Multi-view inpainting, segmentation and video blending, for more versatile Image Based Rendering

Theo Thonat

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Complétion d'image, segmentation et mixture de vidéos dans un contexte multi-vue, pour un rendu basé image plus polyvalent

Théo Thonat

Inria Sophia Antipolis-Méditerannée

Présentée en vue de l'obtention du grade de docteur en Informatique de l'Université Côte d'Azur
Dirigée par : George Drettakis
Soutenue le : 26 Juin 2019

Devant le jury composé de :
Yann Gousseau, Professeur, Télécom ParisTech
Christian Theobalt, Professeur, Max Planck Institute for Informatics
Hendrik P. A. Lensch, Professeur, University of Tübingen
James Tompkin, Professeur, Brown University
Tobias Ritschel, Professeur, University College London
George Drettakis, Directeur de Recherche, Inria Sophia Antipolis - Méditerranée

informatiques mathématiques
RÉGION SUD PROVENCE ALPES CÔTE D'AZUR
AGENCE NATIONALE DE LA RECHERCHE ANR
CR-PLAY Capture Reconstruct Play
erc European Research Council Established by the European Commission

Complétion d'image, segmentation et mixture de vidéos dans un contexte multi-vue, pour un rendu basé image plus polyvalent

---

Multi-view inpainting, segmentation and video blending, for more versatile Image Based Rendering

Jury:

Président du jury / President of the jury
Yann Gousseau, Professeur, Télécom ParisTech

Rapporteurs / Reviewers
Christian Theobalt, Professeur, Max Planck Institute for Informatics
Hendrik P. A. Lensch, Professeur, University of Tübingen

Examinateurs / Examiners
James Tompkin, Professeur, Brown University
Tobias Ritschel, Professeur, University College London

Directeur de thèse / Thesis supervisor
George Drettakis, Directeur de Recherche, Inria Sophia Antipolis - Méditerranée

Acknowledgements

The writing of this thesis and the completion of the work it presents would not have been possible without the involvement of my advisor **George Drettakis**. I want to thank him for his guidance in my research and his ability to provide motivation in moments of doubt.

I would like to acknowledge my different co-authors for their valuable time and resources, without which most of the present work would not have been accomplished. I want to thank in particular **Sylvain Paris** who gave me a successful internship opportunity at Adobe Research, and first recommended me for reviewing.

I am grateful to the members of the different French and European commissions that financially supported my thesis. Thanks also to the anonymous reviewers for their valuable criticism.

I would like to also acknowledge my colleagues at the GraphDeco team who made those thesis years in an engaging company. In particular, I am deeply grateful to many of you for all the help you provided in the crucial moments. Shout out to **Simon Rodriguez**, my partner in crime for clean code and endless graphics related discussions, and to **Johanna Delanoy**, for having faced most of the PhD studies shenanigans together.

I spare a thought the forsaken datasets that required dedication to capture but in the end were unsuitable for my research.

Finally, I want to give a special thank to **Alizée Farshian**, my girlfriend who supported me during those challenging years. The time we spent together was the driving force for my efforts.

Résumé

La création d'images réalistes avec le processus classique de rendu demande un travail manuel considérable, de la génération de modèles 3D à la gestion de l'illumination. Cela demande à la fois des artistes experts modeleurs 3D mais également des machines avec une certaine puissance de calcul.

Se basant uniquement sur des photos prises par un utilisateur lambda, le rendu basé image (IBR) est un moyen alternatif de rendre une scène en temps réel, de manière immersive et réaliste. Ce type de rendu possède des applications dans des domaines tels que le tourisme virtuel, la préservation du patrimoine, la cartographie interactive, la planification urbaine et architecturale, ainsi que la production de films.

De nombreuses méthodes efficaces de rendu base image ont été proposées ces dernières années, mais elles possèdent néanmoins certaines limitations. Tout d'abord, bien que ces méthodes permettent effectivement de générer des images de bonne qualité, il est difficile de pouvoir modifier le contenu de la scène. En effet, la capture d'une scène réelle s'accompagne des contraintes liées a l'environnement au moment de la prise de photos, qui peut ne pas correspondre totalement aux exigences de l'utilisateur.

Ensuite, ces méthodes dépendent grandement de la qualité de la représentation géométrique sous-jacente des scènes. En conséquence, des scènes contenant par exemple des surfaces réflectives, des structures fines ou bien du contenu dynamique, produisent des artefacts visuels important.

Afin de répondre à la première limitation, nous proposons d'étendre la complétion d'image a un contexte multi-vue non structuré, permettant ainsi le retrait d'objet d'une scène. Ce genre de complétion demande non seulement d'halluciner l'apparence, mais également la géométrie de ce qui se trouve derrière l'objet à retirer. Notre méthode réduit les artefacts de rendu en supprimant les objets mal représentés par l'IBR, et permet également de déplacer des objets correctement rendus.

Nous répondons à la deuxième limitation en élargissant le spectre des scènes traitable en IBR, et ce de deux manières. Tout d'abord, nous nous focalisons sur le cas des structures fines qui sont un cas particulièrement compliqué pour la reconstruction multi-vue 3D, et

qui représente une importante limitation pour l'IBR dans un contexte urbain. Nous proposons une méthode qui extrait puis rend les structures fines dont la surface sous-jacente est simple. Nous introduisons un algorithme de segmentation multi-vue pour les structures fines, ainsi qu'une méthode de rendu qui étend le rendu IBR avec de l'information de transparence.

Enfin, nous proposons une première approche pour étendre l'IBR à des contenus dynamiques. En nous focalisant sur des effets dynamiques stochastiques, nous sommes capables de préserver à la fois une acquisition facile à mettre en œuvre et une navigation libre dans la scène rendue. Notre idée principale est d'utiliser une représentation des vidéos adaptée à les mélanger spatio-temporellement et à les faire boucler.

Les résultats de chacune de nos méthodes montrent une amélioration de la qualité visuelle de rendu sur des scènes variées.

Mots-clés: Rendu Basé Image, Multi-vue, Inpainting, Segmentation, Vidéos

---

Abstract

Creating realistic images with the traditional rendering pipeline requires tedious work, starting with complex manual work to create 3D models, materials, and lighting, and then computationally expensive realistic rendering. Such a process requires both skilled artists and significant computing power.

Image Based Rendering (IBR) is an alternative way to create high quality content by only using an unstructured set of photos as input. IBR allows casual users to create and render realistic and immersive scenes in real time, for applications such as virtual tourism, cultural heritage, interactive mapping, urban and architecture planning, and movie production. Existing IBR methods produce generally good image quality, but still suffer from limitations. First, many types of scene content produce visually-unappealing rendering artifacts, because the underlying scene representation is insufficient, e.g, for reflective surfaces, thin structures, and dynamic content. Second, scenes are often captured with real- world constraints which require editing to meet the user requirements, yet existing IBR methods do not allow this.

To address editing, we propose to extend single image inpainting to allow sparse multi-view object removal. Such inpainting requires to hallucinating both color and geometry behind the object to be removed in a multi-view coherent fashion. Our method reduces rendering artifacts by removing objects which are not well represented by IBR methods or by moving well represented objects in the scene.

To address rendering quality, we enlarge the scope of casual IBR in two different ways. First we deal with the case of thin structures, which are extremely challenging for multi-view 3D reconstruction and represent a major limitation for IBR in an urban context. We propose a pipeline which locates and renders thin structures supported by simple surfaces. We introduce both a multi-view segmentation algorithm for thin structures, and a rendering method which extends traditional IBR with transparency information. Second, we propose an approach to extend IBR to dynamic contents. By focusing on time-dependent stochastic textures, we preserve both the casual capture setup and the free-viewpoint navigation of the rendered scene. Our key insight is to use a video rep-

resentation which is adapted to video looping and spatio-temporal blending.

Our results for all methods show improved visual quality compared to previous solutions on a variety of input scenes.

Keywords: Image Based Rendering, Multi-view, Inpainting, Segmentation, Video

---

Table of contents

Chapter 1

Introduction

Contents

1.1 Rendering context

Computer graphics has produced computer-generated images for the past 50 years. A major component is the last step of the image synthesis process, called *rendering*. As means for displaying 2D images, it has many fields of applications such as movie production, graphic design, video games, and medical imaging. From a model of a *scene*, rendering algorithms visualize its appearance from a virtual eye named *viewpoint*. The development of computer graphics techniques has led to more and more complex scenes representations, enabling remarkable progress towards realistic rendering. A scene model usually contains 3D geometry, material proprieties such as textures, normal maps, or bidirectional reflectance distribution functions, lighting conditions, and possibly animation parameters for dynamic scenes. The goal of a rendering algorithm is to compute how light travels from light sources to the desired viewpoint, while interacting with the geometry in between.

Figure 1.1: **Realistic rendering**. Left: Offline rendering from the *Lion King (2019)* movie trailer[1]. Right: Real time rendering from the video game *Battlefield 1* [2].

The two most well-known types of rendering algorithm are *rasterization* and *ray casting*. Rasterization projects the 3D scene onto the 2D screen, and then computes light interactions independently per output *pixel*. This method, benefiting from dedicated hardware accelerators called graphics processing units (GPUs), is suited for real-time applications such as video games. The time budget to compute a single frame is related to the display framerate, typically 60Hz. Ray casting, on the other hand, often uses *global illumination* techniques to simulate the light behavior using physically-accurate models. It works by shooting rays through the scene, to gather contribution of light paths connecting both

[1] https://www.youtube.com/watch?v=7TavVZMewpY/
[2] https://battlefield.fandom.com/wiki/Argonne_Forest

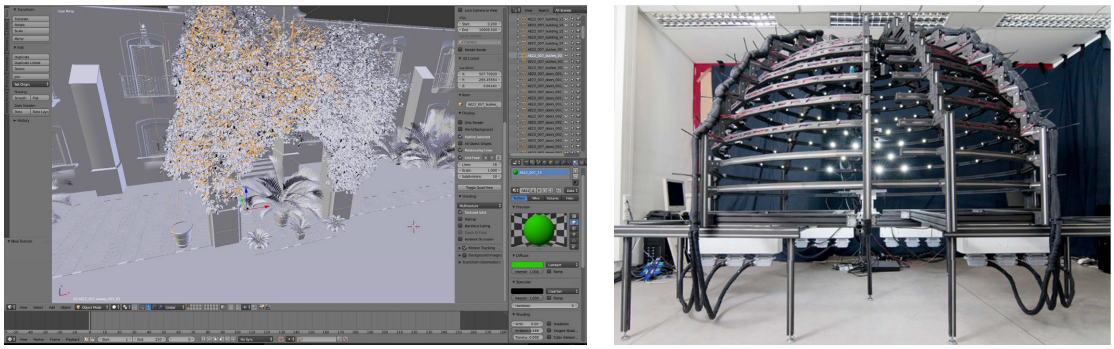


Figure 1.2: **Content creation**. Left: Interface of the open-source 3D computer graphics software toolset Blender[4]. Right: Real-world material appearance acquisition setup of Schwartz et al. [107].

the eye and the lights. This method, computationally more expensive than rasterization, provides a level of visual quality needed for the movie industry. Frames may require hours to be generated, so this offline computing can typically be divided between multiple computers.

Both kinds of rendering algorithm can achieve a certain degree of realism (Figure 1.1). However, this realism relies heavily on the complexity of the scene model. Fine image details require 3D geometry with millions of polygons and textures with high resolution. Realistic illumination is impossible without carefully-chosen materials and lighting setups. Plausible motions require either intensive physics simulations or complex precomputed animations. For example, the amount of data necessary to render the *Island Scene* from the Disney movie *Moana* is about 200GB once unpacked[3].

One of the main challenges of computer graphics is how to obtain or create the realistic content needed for the high visual quality renderings of video games or movies. For example video games might model entire open worlds in all their diversity and complexity. Progress in computer graphics modeling applications has made possible the creation and the management of complex scenes (Figure 1.2, left). However, obtaining detailed geometries or realistic materials require tedious manual work by skilled artists. As a scene is made up of many different components, achieving realism also implies those artists are specialized with a high level of expertise.

[3] https://www.technology.disneyanimation.com/islandscene/
[4] https://www.blender.org/

1.2 Content creation

One way to ease this content creation task is to *capture* scenes from the real world through the measurement of its 3D geometry and material proprieties.

Many technologies exist to capture the geometry of a scene. For example, precise measurements using 3D laser scanning, or *LIDAR*, enable high quality surface acquisition [71]. Yet the capture setup is usually constrained and the dedicated hardware is not affordable for the casual user. Cheaper depth sensors such as the Kinect provide solutions in indoors setup [116]. However, the captured depth is noisy and with low resolution, and often needs manual refinement to be used for rendering. On the contrary, methods using *photogrammetry* are solely based on pictures of the scene. By combining information from multiple images, they provide a way to capture geometry and texture for the casual user, for example using nowadays ubiquitous smartphones [85].

Because of the high dimensionality of material representations, their capture requires more involved setups. Progress has been made to increase the quality of the captured real-world materials, but such techniques typically require laboratory environment (Figure 1.2, right). Recent methods showed that plausible materials could be retrieved from single-shot capture [31]. However, their captured material might be insufficient for high quality realism.

In summary, content creation for rendering is the challenge of building scene representations which are expressive enough to allow realistic rendering, while requiring as few resources as possible. This trade-off is important: for example, in video games or movie production where prototyping with conclusive rendering quality is crucial to allow fast development cycles. Therefore, *images* have emerged as a useful acquisition medium, with an easy capture setup.

Moreover, realism in traditional rendering is typically achieved by considering the *rendering equation* which models global light interactions. Algorithms solving this equation are very computationally intensive and have difficulty achieving real-time rendering. But what is captured when shooting photographs is the real world appearance of objects, meaning the light information is not bound by the model chosen to describe geometry or materials. Therefore, images are suited for realistic rendering in the sense that they already contain the lighting effects that need to be reproduced.

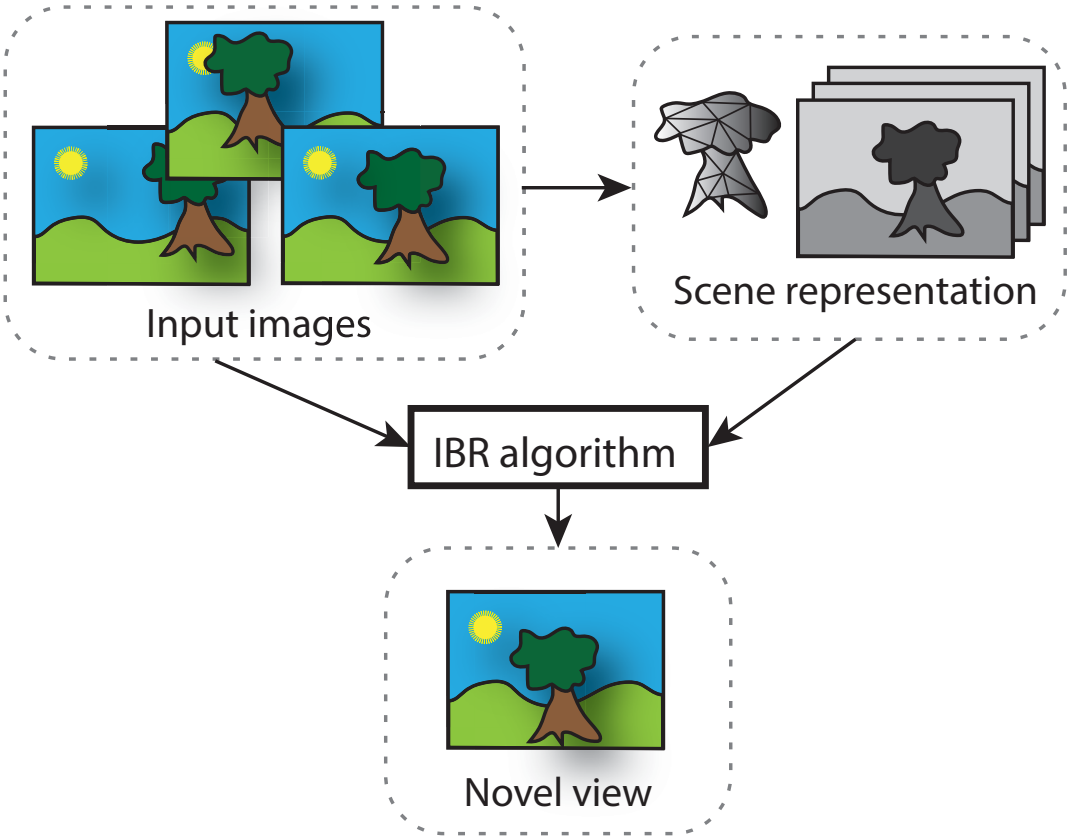


Figure 1.3: **Image Based Rendering principles**. Starting from a set of images, a representation of the scene is computed (here a 3D mesh and depth maps). The IBR algorithm renders novel viewpoints using both the images and the scene representation.

In this thesis, we will focus on Image-Based Rendering (IBR) approaches which avoid complex lighting computations and tedious content creation. We will propose solutions to overcome some of their limitations, namely the lack of scene editability and the narrow scope of scenes types that can be represented by IBR.

1.3 Thesis scope

The key idea behind IBR is that all the complex light interactions are "baked into" the input photos, since nature spontaneously solves physics equations with ideal models, i.e., with the real geometry, materials, and lighting. Therefore the challenge of IBR algorithms is to correctly sample this information to synthesize new viewpoints. More specifically, it comes down to finding *where* in space these interactions happen and *how*

to combine the different sources of information. Therefore, an IBR method relies on a scene representation suited to render novel viewpoints (Figure 1.3). From this general formulation, we will see in chapter 2 that many directions of research have been explored. In this thesis, we will focus on *unstructured* and *wide-baseline* capture setups, with an objective of real-time rendering. This corresponds to a typical scenario of casual capture to provide free-viewpoint immersive scene exploration, which is one of the long-term goals of *Virtual Reality*. One well-known, albeit simple, example of such IBR application is Google *Street View*, where the user can jump from discrete locations to another, to explore urban environments.

As an alternative to the traditional graphics pipeline, IBR rendering methods aim to combine the best of many graphics components such as ease of capture, realism and real time rendering. However they suffer from several limitations.

Manually created graphic assets are perfect by construction, as there is no reference for them to be compared to. On the other hand, a scene model is only an approximation of the captured real scene. As a consequence, IBR algorithms have to deal with imperfect data, such as camera misalignment or approximate 3D geometry. For challenging scenes, the geometry can be erroneous enough to cause rendering artifacts, which are visually unappealing and break immersion. Such challenging cases include vegetation, reflective or transparent surfaces, and thin structures.

Geometry retrieval from images is typically based on *photo-consistency*, which means that a single object should have similar appearance when observed from different views. This assumption is partially broken with a reflective or semi-transparent surface. This is also the case for dynamic scenes, where any change of shape, color, or pose over time produces non-photo-consistent content.

As a result, many IBR techniques are limited to scenes with diffuse materials and “not too thin” shapes, with “still life” capture.

Finally, one other major limitation is the lack of editability of an IBR scene. Indeed, it is important for graphic artists to have control over their scenes, as user wishes are likely to change over time. However, an IBR scene is captured in the real world at a specific moment in time, which might not correspond to the requirements of the user. The illumination might not be the desired one, or there might be clutter in the input photos that should be removed. Therefore there is a need for edition of IBR scenes.

Many methods exists to edit and composite images or dense multi-view datasets, but few have tackled the issue of consistency when dealing with sparse multi-view datasets.

Given those limitations, we will present in this thesis new approaches to improve both the editability and the scope of target scenario for IBR. More specifically, we address three main challenges in three research projects:

- Multi-view image inpainting for IBR scenes. We propose in chapter 3 to extend single image inpainting to a sparse multi-view context, allowing object removal. Such inpainting requires to hallucinate both color and geometry behind the object to be removed, in a multi-view coherent fashion.
- Thin structures in IBR. We enlarge the scope of casual IBR in chapter 4 by dealing with the case of thin structures. Such structures are extremely challenging for multi-view 3D reconstruction and represent a major limitation for IBR in an urban context.
- Dynamic IBR scenes. We propose in chapter 5 a first approach to extend IBR to dynamic contents. By focusing on time-dependent stochastic textures, we preserve both the casual capture setup and the free-viewpoint navigation in the rendered scene, while adding more lively effects to the static IBR scenes.

1.4 Funding

The work in this thesis was funded by a Provence Côte d'Azur Ph.D. Scholarship, the ANR project SEMAPOLIS (ANR-13-CORD-0003)[5], the FP7 EU project CR-PLAY (ICT-611089)[6] and the ERC Advanced Grant No. 788065 FUNGRAPH[7]. The video blending work was partially completed when the author was an intern at Adobe Research.

1.5 Thesis publications

The research projects completed for this thesis have led to two publications in international conferences and journals, and one submission in preparation:

[5] https://project.inria.fr/semapolis/
[6] http://www.cr-play.eu/
[7] http://fungraph.inria.fr

- Theo Thonat, Eli Shechtman, Sylvain Paris, and George Drettakis. "Multi-view inpainting for image-based scene editing and rendering". In: *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE. 2016, pp. 351–359 [123].
- Theo Thonat, Abdelaziz Djelouah, Fredo Durand, and George Drettakis. "Thin structures in image based rendering". In: *Computer Graphics Forum*. Vol. 37. 4. Wiley Online Library. 2018, pp. 107–118 [122]
- Theo Thonat, Yağiz Aksoy, Sylvain Paris, Miika Aittala, Fredo Durand, and George Drettakis. "Practical Video-Based Rendering of Dynamic Stationary Environments from Unsynchronized Inputs". In preparation. 2019

Chapter 2

Previous work

Contents

Image-Based Rendering (IBR) has been an active research field for the past three decades and many different methods have been proposed. A few surveys have been presented by Shum and Kang [110], Zhang and Chen [138] and Shum et al. [111]. They classified IBR methods considering the amount of geometry needed for representing the scene. This spectrum is also inversely related to the number of input images. Methods with many input images require little or no geometry while methods with less geometry require explicit geometry. However, the proliferation of IBR algorithms in the last 20 years no longer fit in a single axis classification.

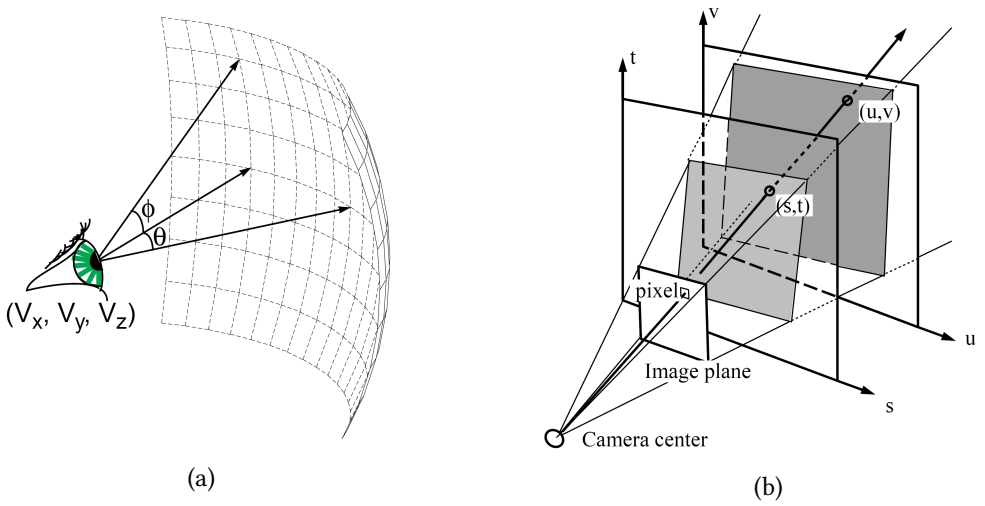
For example, the input data capture setup is crucial to design an IBR algorithm. Some methods require complex setups, with possibly dedicated hardware, while some others only requires a handful of photographs from a casual user. One other criterion to classify IBR methods is the navigation capability. There is always a trade-off between rendering quality and allowing too much freedom of movement to the user. Some algorithms focus on novel viewpoints close to the input views, while others allow extrapolation far from them. It is also possible to restrict the novel view position to allow more angular freedom.

Real world scenes are complex and diverse. As IBR methods aim to capture such scenes with limited data, they have constraints on the type of physical objects they can represent and render. IBR methods can also be organized depending on how general they are, or how far they can push rendering quality for certain types of scenes.

Finally, the type of application of an IBR method influences computation time. Virtual world exploration requires interactivity and real time rendering. The constraints are not the same when designing IBR methods as an offline rendering process.

2.1 Early methods

To describe a scene from a human vision perspective and in the most generic way, Adelson and Bergen [2] introduced the plenoptic function $P(V_x, V_y, V_z, \theta, \phi, \lambda, t)$. This seven-parameter function measures the light intensity perceived by an omniscient observer from any position in space (V_x, V_y, V_z) and in any direction (θ, ϕ), for a given wavelength λ and at a time t. In the most common IBR setups, we only consider static scenes which do not depend on time. Moreover, we only consider fixed wavelengths corresponding to the input images color space, typically three for the red, green and blue channels. The plenoptic function in our context can then be simplified to $P(V_x, V_y, V_z, \theta, \phi)$. The goal of



(a) (b)

Figure 2.1: **(a) Plenoptic eye**. In standard IBR setups, the plenoptic eye represents what a camera captures at a position (V_x, V_y, V_z) from all the possible directions (θ, ϕ). Figure from [83]. **(b)** 4D parametrizations of the plenoptic function using two planes, as described in [44].

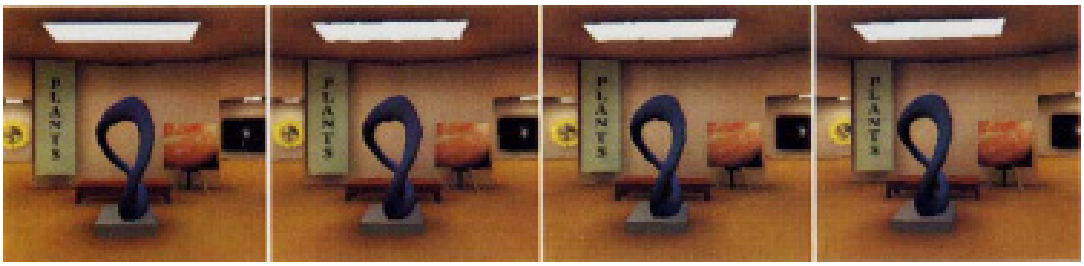


Figure 2.2: **Interpolation for view synthesis.** Given the images at both ends, the method of Chen and Williams [24] uses morphing, based on image correspondences, to generate the two intermediate views in the middle.

any Image-Based Rendering method, which typically is to synthesize a novel viewpoint, then consists of simply sampling this function with the desired parameters.

Unfortunately, considering its large dimensionality and the continuous nature of its parameters, this ideal representation cannot be completely known for a real-world scene. However, the set of images from a scene, which is the input of any IBR method, provides direct samples of the plenoptic function.

Early novel view synthesis methods, by Chen and Williams [24] and Laveau and Faugeras [67] demonstrated that in the case of synthetic data, a set of photographs can be an inter-

mediate representation of a scene. Seitz and Dyer [109] investigated the class of scenes for which image warping is theoretically a physically valid mechanism for view interpolation, pointing out the fundamental issue of occlusions. Chen [23] presented a system to navigate in a virtual environment using only real images. By computing panoramic images, the system allows to jump from one viewpoint to another. It reduced the initial complexity of the plenoptic function by considering 2D cylindrical (θ, ϕ) panoramas at a discrete set of 3D positions (V_x, V_y, V_z). The generation of such panoramas was improved by Szeliski and Shum [119], allowing as input only a mosaic of unstructured images.

The first complete IBR methods able to produce continuously novel viewpoints from input photos were proposed by McMillan and Bishop [83], Levoy and Hanrahan [70], and Gortler et al. [44]. They tackled the view interpolation problem by resampling the plenoptic function. Considering an occlusion free setup for capturing a single object, and limiting the novel viewpoint to be outside of the object convex hull, these methods can reduce the plenoptic function to a 4D parameterization using two parallel planes. Levoy and Hanrahan [70] introduced the Light Field capture setup for rendering, where a large set of input cameras are placed regularly on a 2D grid. Gortler et al. [44] introduced the Lumigraph, where a large set of photos can be taken in a semi-unstructured fashion in a sphere around the object. They also showed how approximate geometric information could be used to chose the right plenoptic resampling kernels. Both methods require around a thousand images for a single object. This data redundancy makes these methods particularly well adapted for highly specular objects and makes them able to provide coherent results even without the need for explicit geometry.

Even though these approaches make good use of the theoretical plenoptic framework, their constrained capture setup restricts in practice the targeted scene scope to a single indoor object.

Other methods rely on geometric information to provide navigation for novel view synthesis. Debevec et al. [29] use user-assisted photogrammetry to obtain a set of polygons to describe the scene in an architectural context. This geometry allows to reproject the textures from the input views onto the novel view, taking in account visibility. Multiple images might overlap when reprojected. Since they are not guaranteed to be similar because of non Lambertian surfaces, they must be blended. Therefore, they introduced the fundamental idea of blending images according to the novel view point and the ge-

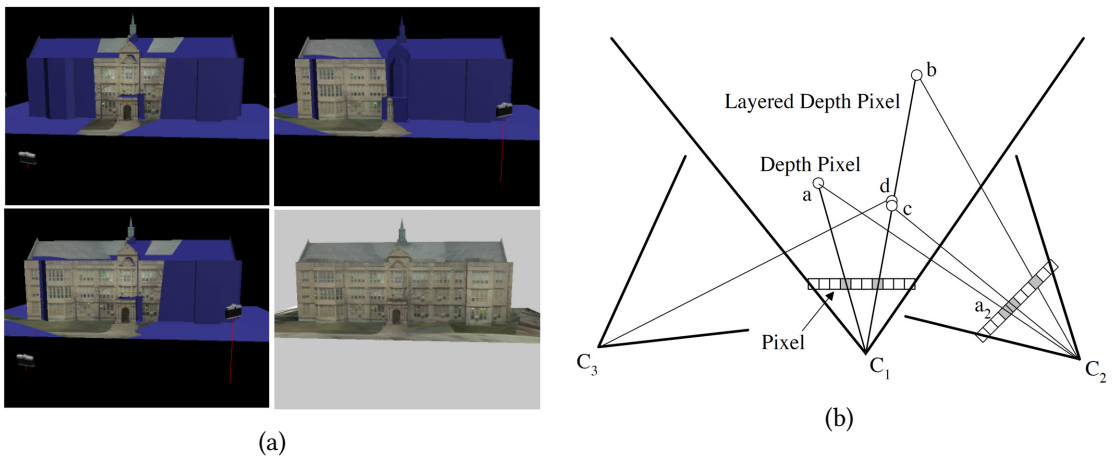


Figure 2.3: **(a) View Dependent Texture Mapping**. Projection on a geometry (blue) of photo textures from calibrated cameras, with each camera covering the part it sees (three first images). VDTM from Debevec et al. [29] blends textures from multiples viewpoints whenever they overlap (last image). **(b) Layered Depth Image** from He et al. [51]. Given a depth of a pixel, correspondences with pixels in other views are possible. Moreover, in case of semi-transparent objects, multiple depths can be assigned to a single pixel.

ometry, called View-Dependent Texture Mapping (VDTM). Debevec et al. [30] improved this method by precomputing for each polygon the texture information available in the three most relevant input views. This approach allows also fills non-visible polygons in object space by diffusing colors from surrounding visible polygons in a preprocess step.

Instead of relying on a global geometry, He et al. [51] use a per-view layered representation of both sprites and dense depth-maps, called Layered Depth Images. Such a representation provides better occlusion handling thanks to the multiple depth information per pixel. Lischinski and Rappoport [75] extended this work by tackling the case of non-diffuse synthetic scenes. Both view-dependent and view-independent appearances are extracted into two Layered Depth Images, which can be rendered separately before compositing.

By studying the plenoptic function in the spectral domain, Chai et al. [20] extracted analytical bounds for light field sampling. They introduced the notion of minimum sampling curve in IBR, describing the relationship between depth information and number of image samples given an output resolution.

All the previously mentioned methods have strong limitations, either requiring specific

capture setups focused on one object [70, 44], limited exploration capabilities [23], or involved manual interaction [29, 51]. In the next section, we will see how breakthroughs in computer vision made possible IBR methods at a different scale in terms of scene complexity.

2.2 Geometry based IBR

Representing the scene geometry by a 3D model, called a *proxy*, has many advantages for IBR. First, such geometry naturally provides dense correspondences between input images. Secondly, it is suited for real time IBR as rasterization of the proxy for any novel viewpoint can be performed on the GPU. Finally, it enables easy occlusion handling by comparing renderings of the proxy from the input views and the novel view. However, automatically obtaining a proxy purely from a set of input images is a difficult task. This is called multi-view stereo reconstruction or *photogrammetry*. We will first describe the reconstruction process and then explain how it impacts IBR algorithms.

2.2.1 3D reconstruction

Obtaining a 3D model of a scene from a set of input photos requires two main steps. First, we must find a common 3D space for the input cameras, and then we must fill this space with dense appearance information.

The first step, called Structure from Motion (SfM), estimates the parameters of the camera models associated to the input photos. It is based on 2D correspondences between two or more images. Finding such correspondences requires suitable image descriptors able to represent the same 3D feature seen from different viewpoints. Lowe et al. [81] and Bay et al. [10] introduced respectively SIFT and SURF, two local image descriptors with scale and rotation invariance.

This set of 2D correspondences can then be used to estimate intrinsic and extrinsic cameras parameters, which we call *camera calibration*. This also provides the 3D location of all the correspondences, which is considered a sparse reconstruction. SfM methods became renown with the work of Snavely et al. [117], who presented a system able to calibrate up to thousands of images gathered from online search. Their iterative algorithm minimizes the reprojection error between the 2D feature points using a *bundle adjustment* optimization.

(a)

(b)

Figure 2.4: **(a) Structure from Motion**. Calibration of 21K photos (in red) using COLMAP [105]. 2D correspondences are used to create a sparse colored MVS 3D point cloud. **(b) Multi-View Stereo**. Dense 3D reconstruction using the method of Jancosek and Pajdla [58], which automatically provides an approximate geometry of a scene.

The second step, called Multi-View Stereo (MVS), estimates the dense 3D profile of the scene. Methods such as the one proposed by Pons et al. [98] obtain a proxy using a variational formulation based on a global measure of image similarity. It handles more appearance variation due to the modeling of non-Lambertian materials. However this method scales poorly with the number of images, dealing with typically less than a hundred input photos. Other methods have a more traditional approach to dense geometry reconstruction. They estimate dense depths maps from the calibration and the image content. A final step, called fusion, combines the information from all depths map into a single 3D mesh. For a given pixel from a given view, this step looks for the best matches along the epipolar lines in neighboring views.

Methods robust to occlusions, difference in lighting or scale, and dealing with a large number of images have been developed, for example by Goesele et al. [43] using normal-based photo-consistency. Using global visibility constraints, the method of Furukawa et al. [38] can handle large datasets: over ten thousands images for a single scene. Relying entirely on photo-consistency, previous methods had issues with large textureless regions. Jancosek and Pajdla [58] used a visual-hull approach to provide plausible reconstruction for these weakly supported surfaces. It provides approximate 3D even for non Lambertian surfaces, enabling the reconstruction of a wider set of practical scenes.

Existing reconstruction software packages provide both calibration and dense stereo components. One of the most popular is Visual SfM[1], using the SfM method of Wu [131], combined with either the methods of Goesele et al. [43] or Furukawa et al. [38] for the dense reconstruction. More recently, the state of the art for open software general purpose 3D reconstruction is COLMAP[2], based on the work of Schönberger and Frahm [105] for SfM and Schönberger et al. [106] for MVS. Another state of the art pipeline is the proprietary software Reality Capture[3] [101], based on the work of Jancosek and Pajdla [58].

2.2.2 Reprojection based IBR

The ability to automatically calibrate cameras and produce a 3D representation of a scene is an important source of information for IBR methods. It provides approximate correspondence between the views, which enables the capacity of blending different image content for the same 3D location. It also provides a way to select the relevant input camera associated to novel view to render, enabling better scalability for IBR scenes.

The first method to describe what is now the standard free-viewpoint navigation IBR pipeline was proposed by Heigl et al. [54]. The first step is to obtain calibrated input views and global proxy 3D geometry via SfM and MVS. The second step is a custom real time VDTM rendering algorithm to select cameras, reproject input views, and blend them. In the case of Heigl et al., they apply this pipeline to hand-held video sequences, using local and global planes to guide the color mapping.

Given a simple 3D geometry, Buehler et al. [14] proposed a general purpose IBR method,

[1] http://ccwu.me/vsfm/
[2] https://colmap.github.io/
[3] https://www.capturingreality.com/

Figure 2.5: **Reprojection Based IBR**. From a set of wide-baseline captured images (top: 3 out of 50 input images), MVS methods are able to produce approximate geometry in a preprocessing step (bottom left). At runtime, the geometry is used to compute the contribution of each input image thanks to a blending obtained from the method of Buehler et al. [14] (middle bottom: a color is associated to each input camera). The input images are reprojected onto the geometry and blended accordingly, producing the final result (bottom right).

extending both Light Field Rendering and the Lumigraph. They derived a set of desired properties for IBR methods and suitable view-dependent mapping blending weights to satisfy them. Such proprieties include epipolar consistency, equivalent ray consistency, resolution sensitivity, continuity, and minimal angular deviation. Epipolar and ray consistency ensure that rays shared by the novel view and the views should be used accordingly. The resolution sensitivity makes sure that the sampling resolution of the reprojected input texture matches the output rendering. Minimal angular deviation defines the metric for camera closeness: two cameras are considered close to each other with respect to a 3D point if the angle between the point and the cameras center is small. Finally, the blending weights must be piecewise smooth with respect to the novel view position, such that any change due to navigation does not produce seam artifacts. As a consequence, a piecewise smooth continuous blending field is computed at each frame, describing for each output pixel the contribution of the input cameras.

An example of this pipeline is illustrated on Figure 2.5. This framework allows complete freedom in the navigation, deals well with diffuse and relatively specular surfaces. However, it is very sensible to poor camera calibration and uncertain geometry near occlusion boundaries. As a consequence, textures can be misaligned or be wrongly reprojected in

an incorrect fashion, causing ghosting artifacts.

Because view-dependent texture mapping methods rely on an approximate geometry, the projection of the input images might not be consistent enough, producing blurring and ghosting artifacts. To alleviate this issue, Eisemann et al. [33] introduced floating textures, where the reprojected images are warped locally based on the *optical flow* between non-occluded regions. They also identify that approximate geometry leads to approximate visibility information, which produces local but visible incorrect boundary occlusions. Therefore, they proposed a soft-visibility scheme where the pixel contribution is related to the distance to its nearest occlusion edge.

The approximate geometry from MVS leads to some pixels with uncertain and with undefined depth. By randomly sampling uncertain pixels along their epipolar lines, the method of Goesele et al. [42] created a data structure called Ambient Point Cloud. Combined with standard colored proxy, it can produce plausible transitions between input views.

These methods used a global representation to describe the scene, enabling true free-viewpoint navigation. The heuristics used to combine color information from different views can compensate for most geometry approximations. In this regard, these methods demonstrated that wide baseline IBR was a powerful tool to render real captured scenes. However, they are very sensitive to incorrect geometry and visibility information. Thus, many challenging scenes containing complex geometry such as vegetation or thin structures are impossible to render without severe visual artifacts. We will see that subsequent methods deal with this issue by increasing the quality of the geometry representation, focusing first on the global proxy, and later focusing on more image-aware representations.

2.2.3 Improving the global proxy

Since multi-view stereo algorithms try to be as general as possible, their trade-offs regarding global consistency versus accuracy might not be suited for IBR applications. This is the reason why some methods decide to compute a global geometry on a smaller range of target scenes with a representation less prone to rendering artifacts.

In an urban context, Furukawa et al. [37] proposed a method to extract orthogonal planes supported by the scene's dominant axis, to handle the case of texture-less image regions.

Figure 2.6: **Super-pixel over-segmentation**. To hallucinate depth in poorly reconstructed regions, Chaurasia et al. [21] use over-segmentation to propagate depth information. To create a reliable depth for the red super-pixel, 3 matches (cyan) are selected from a set of similar and close candidates (yellow).

Sinha et al. [114] proposed a more general pipeline where lines are extracted in the scene thanks to detection of vanishing points. Candidate planar structures are then fitted from the lines. The global geometry is obtained by solving a labeling problem, where pixels need to be coherently assigned a plane, based on photo-consistency.

The method of Gallup et al. [40] extended previous methods by combining standard reconstruction and planar surfaces extraction when appropriate. The segmentation between planar surfaces and non-planar surfaces is performed using both photo-consistency and a learned classifier trained on manually annotated data.

2.2.4 Improving per view geometry

The global geometry obtained from multi-view stereo is a representation which tries to be as consistent as possible with the input view content. However, this consistency implies local losses on the accuracy of the input image-associated depths. This is why several IBR methods decide to use a per view representation, which allows the geometry information to match more closely to the image content.

In the context of view synthesis for interpolating synchronized video frames with small baseline, Zitnick et al. [140] introduced *super-pixel* oversegmention for more robust and edge-aware stereo. They extended the layered depth image representation by also computing extra layers for alpha matting at the occlusion boundaries. Such a representation allows precise contouring and reduces blending artifacts at the occlusion boundaries.

In the same spirit of silhouette handling, Chaurasia et al. [22] rely on user input to identify relevant sparse depth constraints to guide silhouette-aware warps. Thanks to elastic bands between the silhouettes, these local warps reduce the image distortion while keeping precise occlusions boundaries in a wide-baseline setup. To remove the dependency on user interaction, the method of Chaurasia et al. [21] uses automatic image super-pixel oversegmentation from the popular algorithm of Achanta et al. [1]. Combined with MVS point information, their method identifies poorly reconstructed super-pixels and hallucinates their depth by propagation from similar neighbor super-pixels. Assuming local planarity, each super-pixel is reprojected at run-time onto the novel view using rigid shape-preserving warps.

The method of Ortiz-Cayon et al. [90] also improves the per-view scene representation. Instead of estimating the correct per-view geometry, they estimate, from a set of fixed IBR methods, which method has the best rendering quality. A leave-one-out strategy is used for this purpose, where one input image is taken as ground-truth and is compared to the rendering from the same viewpoint using only the rest of the dataset. Using a Bayesian formulation, this meta-IBR algorithm provides, on a per-view and per super-pixel basis, what input IBR method to choose at run-time. One application is the possibility to know, given IBR methods with different computing power requirements, where there is no significant loss in rendering quality to apply the cheapest one.

A few methods rely on a proxy representation which combines multiple representations. Lipski et al. [74] use an hybrid approach combining image morphing thanks to optical-flow and per-view geometry. Such a scheme allows to estimate the two sub representations jointly benefiting from the strengths of both. They demonstrate applications such as stabilization or compositing in the case of free-viewpoint video. Using as input frames from a RGB-D video, the method of Hedman et al. [53] use a representation that combines a rough global proxy as well as per-view geometries. The global geometry provides the possibility of fuzzy depth tests while the per-view meshes allow precise depth contours. They also have a scene partitioning scheme in tiles providing scalable performances even with a large number of images. Starting with initial per-view depth maps from MVS, Penner and Zhang [95] compute a per-view volume modeling the depth distribution uncertainty. This discretized representation provides soft estimation of visibility, allowing an occlusion aware depth estimation. They demonstrate compelling view interpolation results for both wide baseline and dense capture.

Figure 2.7: **Gradient Domain IBR**. By estimating depth at strong image edges, the method of Kopf et al. [64] reprojects image gradients from both real objects and objects reflected by planar surfaces.

Methods relying on per-view representations typically require a lot of preprocess computation, a large memory consumption at runtime, and computationally intensive rendering. However, their success show that accurate visibility via depth estimation near edges is decisive for high quality IBR. By relaxing the general global proxy MVS reconstruction constraints, these methods provide a scene representation more suitable for IBR.

2.2.5 Reflective and semi-transparent surfaces

Previous methods used different ways to represent the scene geometry. However, they all followed the assumption of opaque materials, which means that a pixel can be described by only a single depth. Such a hypothesis is broken in the case of semi-transparent or specular surfaces. For these challenging cases, a pixel encodes information from multiple depths and so free-viewpoint navigation implies some kind of multi-depth stereo combined with image layer separation.

The method of Sinha et al. [113] handles reflective and glossy surfaces in unstructured real images. By modeling the scene with piece-wise planar layers, they assume that virtual objects are associated to reflective layers. Instead of looking for only one minimum in the stereo matching cost, they look for possibly two minima in the case of strong reflections, leading to multiple depth maps, reflective layers, and a reflective fraction map. Standard textures reprojection can then be applied on the layers separately before compositing.

To deal with the case of reflective surfaces, Kopf et al. [64] proposed to treat the novel

view synthesis problem in the gradient domain. Their key observation is that parallax is supported mainly by the image gradients, so they only need to estimate depth at pixels with non-negligible gradient magnitudes. At run time image gradients are reprojected according to their corresponding depth, and the novel view is obtained by solving a Poisson equation. Without boundary conditions, such a problem needs a regularization obtained by a weakly weighted data term. They also provided heuristics to detect occlusion boundaries in the gradient domain.

In the case of synthetic data, Lochmann et al. [80] proposed a method to handle non planar reflections and refractions. As there is no explicit mapping between what two inputs views observe through a curved surface, they rely on a per pixel optimization. Their method uses a light path data structure to provide initial guesses, allowing real time novel view synthesis.

Geometry-based methods have been proven to be a powerful representation for IBR, enabling scalability, high visual quality and real time rendering. However, they strongly rely on the quality of the reconstructed 3D geometry. Recent IBR algorithms improved this representation by relaxing global consistency and focusing on per-view data structures more suitable to the input image content. However, many real world scenes are complex and challenging to be represented and rendered.

In this thesis, we will show how the use of suitable algorithms can produce a more versatile IBR, able to handle a wider scope of scenes with also more control over them.

2.3 Optimization based IBR

Most of the IBR methods described so far focus on finding the most suitable representation and data structure for the scene which would allow real time exploration with visually-appealing quality. However, instead of trying to solve the issue of the intermediate scene representation, others algorithms were proposed to solve the view synthesis problem directly. These methods provide their own problem formulation adapted to their applications. Synthesizing a novel view then typically requires solving a global optimization.

Figure 2.8: **Hyperlapse**. In order to stabilize video shots, Kopf et al. [63] find a suitable camera path for the novel view (left). To produce the final renderings, input video frames are stitched and blended using an offline global optimization (right).

2.3.1 Offline IBR

Fitzgibbon et al. [35] formulated IBR as finding the most likely novel view in a Bayesian framework. Multiples priors are required to define an objective function. These priors include a geometry photo-consistency constraint and texture based prior to guide the solution towards a result with similar texture statistics as the inputs images. To generate smooth time-lapse from an first person video with irregular motion, Kopf et al. [63] designed an approach where the objective is to find the most visually appealing camera path in the scene. Their method combines two sub optimizations. The first one uses calibration information to find a smooth camera path following the input video, while ensuring full rendering coverage for the output. The second one is a labeling problem where each pixel of the output has to be assigned the input camera from which it should take its color information from. Concerning the data term for the labeling optimization, they proposed that many heuristics from View Dependent Texture Mapping (VDTM) could be superseded by a minimal texture stretch criterion based on the Jacobian of the textures coordinates.

The Bayesian framework of Pujades et al. [100] proposed a unified formulation, which allows the derivation of most of the heuristics from VDTM. Their data term models the texture deformation induced by the proxy based reprojection. It also models the uncertainty in both geometry from reconstruction and color from the sensor. Their prior term is a convex total variation regularizer. Even though their formulation is very general, the underlying optimization is only guaranteed to converge towards a local minimum. Moreover, it is computationally expensive to solve and requires several seconds to gen-

erate a single novel view.

By ignoring the real-time constraints, offline IBR methods are able to model the novel view synthesis problem as global optimizations. However, the more general formulations proposed only provide theoretical justifications for the previous methods heuristics, and did not really overcome their limitations. Moreover, the manual design of some of these formulations make them adapted only to specific tasks. We will see in the next section how recent deep learning approaches are able to alleviate the issue of the heuristic manual design.

2.3.2 Learning based

Machine learning algorithms, and especially *deep neural networks*, have become an essential tool for computer vision. They have also been proven useful for computer graphics, and in particular for IBR with the work of Flynn et al. [36]. They presented a deep architecture producing view synthesis directly from calibrated input images. It consists of two components dedicated to predict depth and color for the novel view. *Plane-sweep volumes* from neighboring views, encoding epipolar constrains, are fed into the network to learn MVS. While the network is trained only once and shared across scenes, it requires minutes to generate a novel view. It is also limited in output resolution and has blur artifacts.

By focusing on a network which only predicts the selection weights for VDTM with per-view geometry, Hedman et al. [52] built upon the work of Hedman et al. [53] to produce a scalable learning-based IBR method at interactive framerates. To synthesize a novel view, they use a *U-Net* network architecture, combining both a textured mesh rendering and reprojections of neighboring views. This method can partially correct incorrect occlusions and differences in input image illumination. Following the same intuition, Thies et al. [120] also proposed an architecture which uses reprojections from neighbor views using a proxy. However, they focus on the extraction and the regression of view-dependent effects. Their training uses both synthetic and real data in a self-supervised fashion, and relies on the dense capture of a single object.

Instead of optimizing for a blending strategy between reprojected images, Sitzmann et al. [115] proposed an architecture to learn a self-contained representation of an object, which can then be queried to render a novel viewpoint. They use a persistent 3D feature

voxel based representation and an occlusion network which predicts soft-visibility for novel viewpoints. This architecture must be trained for each novel object using hundreds of images. Their method is proxy-free and is able to synthesize novel views with high quality, but the underlying representation has 170 millions parameters for a single object.

2.4 Conclusion

We have seen that the landscape of IBR algorithms is vast, with many research directions. In the past three decades, the IBR approaches which have been proposed rely on different representations of scenes, depending on the application and on the capture setup. Early methods used dense sampling without geometry, but following methods reached scalability and high rendering quality for many use cases thanks to a geometric representation. Recently, other promising representations have been proposed, notably voxel or learning based, but none has yet achieve the goal of producing a virtual environment with convincing realism and flexibility on the navigation, from an *in the wild* casual capture without manual work.

Producing plausible and realistic views is challenging with incomplete or incorrect 3D information. As most IBR methods strongly rely on MVS techniques for the geometric representation, they cannot deal with scenes where acceptable geometry is hard to generate. Such a limitation thus severely impacts the scope of IBR. Moreover, wide-baseline IBR as a means of accessible content creation suffers from the lack of scene editability. Indeed, the sparsity of the input makes any modification in image space challenging to propagate to the whole dataset in a coherent fashion.

In this thesis we address some of these issues by using the most of the available multi-view information. In chapter 3, we will see how extending single image inpainting to a sparse multiview context allows some editability for IBR scenes. In chapter 4, we will see how a muli-view segmentation approach enables the detection and the rendering of thin structures. Finally, we will present in chapter 5 a novel video representation allowing looping and multi-view video blending for time-dependent stochastic textures.

Chapter 3

Multiview inpainting

Contents

3.1 Introduction

We have seen in chapter 2 that recent progress of IBR algorithms [139, 74, 21] allows free-viewpoint navigation in large regions of space. Combined with the massive data-acquistion efforts such as Google Street View or Microsoft Bing, IBR promises to provide the sense of "being there" for almost any location on the globe from within a web browser. However, a major downside of IBR is that it relies on multi-view photo datasets which must either be free of clutter (pedestrians, cars, signposts etc.) at capture time or requires painstaking editing to be usable for IBR.

To edit a multi-view dataset for IBR, changes in both color and depth must be propagated to all views to keep the dataset consistent. Because every photo shows the scene from a different viewpoint, this propagation is challenging, especially when viewpoints are far apart from each other, i.e., in the wide-baseline case which is the focus of our work. Single-image inpainting, e.g., [27, 57], does not solve this problem because it does not ensure consistency between views. Neither does video inpainting, e.g., [129, 61], because it requires dense data, for example to compute optical flow. Furthermore, inpainting for IBR must also infer consistent depth so that parallax can be properly rendered, which none of these techniques support.

In this chapter, we present a semi-automatic solution to multi-view inpainting and editing for IBR. Our algorithm takes as input a set of images and a set of masks to specify what to remove. It inpaints image, normal and depth content coherently across views and consistently with the depth structure of the scene. With our algorithm, one can easily remove passers-by, cars, street signs, and other distractors that typically clutter IBR datasets, enabling the rendering of clean unobstructed views and even limited editing of the scene such as moving isolated objects.

3.2 Previous work

Single Image Inpainting. Criminisi et al. [27] proposed an inpainting algorithm which can retrieve basic image structures, by using a well-chosen filling order, but works better with relatively small regions to inpaint. Barnes et al. [9] developed PatchMatch, which finds approximate nearest neighbor matches between patches using random search. It can be used for inpainting and achieves a speedup of several orders of magnitude over previous work. Huang et al. [57] used planar information to guide the search space for

patch matches, by estimating planar projection parameters and plane segmentation. In contrast, we leverage 3D information from multi-view reconstruction which provides an additional source of data for the patch search in our setting.

Video Completion. Wexler et al. [129] introduced a method for video sequence completion using spatio-temporal patches and a multi scale approach. Newson et al. [86] improved this technique by using an accelerated spatio-temporal search, and by introducing texture features to the patch distance to correctly inpaint video textures. Klose et al. [61] deal with a general sampling-based algorithm for processing applications of a scene's video. The technique first collects a very large set of input video pixels and then filters them iteratively before visually converging. Other methods such as the one proposed by Granados et al. [45] consider video inpainting as a labelling problem, but requires manual tracking of the object to inpaint. Multi-view information has been used to enhance low-resolution videos, for example by Bhat et al. [11]. Video completion and enhancement methods provide important insight and can also be used as a methodological framework for multi-view datasets. The algorithms are nonetheless inherently different to ours since we assume wide-baseline photographs as input.

IBR and Multi-Image methods. Fitzgibbon et al. [35] use a patch-based approach for novel view synthesis in IBR. In contrast to our wide baseline data, they treat small-baseline datasets. In general IBR methods are designed to fill small holes due to depth disocclusions, and do not always adapt well to the more general wide baseline inpainting problem we address.

Graph-cuts have been used when mixing images from different sources , for example by Agarwala et al. [3]. Our approach is different in that we use multi-view reprojection and the associated *confidence* as a guide for patch-based inpainting. The shift-map algorithm by Pritch et al. [99] also uses graph-cut for hole filling, where the labels are image locations, while we will operate on color directly. Darabi et al. [28] extended the patch space search by adding rotated, re-scaled, and photometrically-transformed patches. Multiple images were used, but only as additional sources yielding good quality inpainting.

There has beeen some work using multiple views to remove objects from images. Whyte et al. [130] replace a user-defined target region from a query image using internet photographs of the same scene. Using homographies and photometric registrations, the

method can blend information from the entire dataset to synthesize encouraging results. Hays and Efros [47] use a large database of internet photos for image completion. However, the method is inherently single-image and would not necessarily produce consistent results over a multi-view dataset. There has been plenty of work on RGB-D completion, including attempts to inpaint depth, typically restricted to stereo pairs, as proposed by Howard et al. [56]. In contrast, we target casual, wide-baseline capture, possibly with a mobile phone camera.

In recent work, developed concurrently with ours, Baek et al. [6] proposed a multi-view inpainting method jointly inpainting depth and color. This technique and ours share the same strategy of using depth and reprojected data to guide inpainting, but their scopes differ in major ways. Their method is about image editing, and reconstructs per-image depth maps to handle occlusions, e.g., for inserting an object behind another one. Such depth maps are not sufficient for image-based rendering because they do not provide a consistent 3D representation shared across the images, which is needed for free viewpoint navigation. Our approach specifically addresses this scenario and generates such a global 3D representation.

3.3 Overview

Our method aims to remove objects from wide-baseline multi-vew datasets in a multi-view coherent fashion. To complete holes left in an image by a removed object, we use other views to "see" what is behind the removed object via IBR reprojection, or when such information is not available, e.g., a car big enough to hide a portion of the scene in all views, we use patch synthesis. We describe in section 3.4 a unified approach to combine reprojected content with inpainted content, so that consistent color, normals, and depth are produced across all views. We carefully balance these two sources of information so that inpainting progressively takes over reprojection when multi-view data is less reliable, e.g., coming from distant views or observed at grazing angle.

We explain in section 3.5 how we generate the reprojected content using a graph-cut based global optimization. To extend single image inpainting to a sparse multi-view context, we introduce in section 3.7 a multi-view patch search and multi-view consistent reconstruction method, taking into account the inaccuracies of the approximate 3D reconstruction, while exploiting the global 3D consistency it provides. We also propose

Figure 3.1: **Method overview**. (a) 3D reconstruction from multi-view input image dataset (b) Input images above, input masks below (c) Result of reprojection and mincut, remaining black pixels will be filled through patch-based inpainting (d) Inpainted result.

a multi-view consistent initialization step which is an important element to the success of our approach, detailed in section 3.6.

Importantly, our inpainted multi-view datasets have color and depth consistent with the global 3D reconstruction, allowing the use of the result for IBR. We show two usage scenarios for our approach in section 3.8. The first uses object recognition methods to remove classes of objects (e.g., cars, pedestrians etc.). For the second, we provide a multi-view object removal interface to allow scene editing in a free-viewpoint IBR setting.

3.4 Unified multi-view coherent inpainting algorithm

Our input is a multi-view set of images I_i of a scene, and a set of corresponding masks $\mathcal{M}_i$ which cover the parts of the images we want to remove. Masks are either automatically extracted *axis-aligned bounding rectangles* or regions created with a user-assisted process; we describe mask creation in section 3.8.1. Our goal is to reconstruct images $\mathcal{I}_i$ in which the image content in the mask is removed and replaced by plausible content. We introduce a multi-view inpainting algorithm which builds on IBR and a patch-based algorithm [129], using PatchMatch [9] for search, guided by the multi-view data. A naive approach would be to first reproject as much data as possible from the other views, and use inpainting to fill in the remaining holes where data is missing. However, this simple strategy ignores that not all reprojected data are equally accurate. For instance, data observed at grazing angle are degraded because of foreshortening, and in practice, the 3D reconstruction and camera calibration are not perfect and data coming from distant views are less likely to be accurate.

3.4.1 Multi-view input data

Our typical input consists of 20–40 photos of a scene, with approximately 1.5–2m distance between shots. The images are then calibrated using Structure from Motion [132], and we use CMPMVS [58] for the 3D reconstruction. This first step gives us an approximate mesh and calibrated cameras. Normals and 3D positions are provided approximately by the mesh, but regions not covered by the geometry remain. Several methods exist to propagate depth and/or normals in images; We extend the method of Chaurasia et al. [21] which propagates depth using oversegmentation to also propagate normals; these are precise enough to guide the patch matching process.

3.4.2 Problem formulation

Considering an image I_i that we are inpainting, we obtain data from the other images I_j ($j \neq i$) by reprojecting them into the mask $\mathcal{M}_i$ using the 3D reconstruction and camera calibration. In many cases, not all pixels in $\mathcal{M}_i$ are covered by the reprojected pixels; these correspond to the black pixels in Figure 3.1(c) and in Figure 3.2(left). We write $\mathcal{M}_i^r$ the pixel coordinates in $\mathcal{M}_i$ which have a valid reprojection; $\mathcal{I}_r$ denotes the resulting image. We illustrate an example of $\mathcal{M}$, $\mathcal{M}_r$ and $\mathcal{I}_r$ in Figure 3.2.

Figure 3.2: **Reprojection step**. Left: A crop of an input image showing the mask $\mathcal{M}$ as a dark region. Right: The resulting image $\mathcal{I}_r$ from reprojection into $\mathcal{M}$. The non-black pixels within the box are the region $\mathcal{M}^r$.

The pixels in $\mathcal{M}^r$ contain information from other views with varying accuracy. Where this information is reliable, we guide the patch synthesis to produce pixels similar to those in $\mathcal{M}^r$ by defining the following energy for an input image $\mathcal{I}_i$ with a mask $\mathcal{M}_i$:

$$E(\mathcal{I}_i|\mathcal{M}_i) = \sum_{p\in\mathcal{M}_i} \alpha(\mathcal{I}_r(p) - \mathcal{I}_i(p))^2 + (1-\alpha)\left(E_{\text{col}}(\mathbf{t}_p, \mathbf{s}_q) + E_{\text{3D}}(p)\right) \tag{3.1}$$

where $\mathbf{t}_p, \mathbf{s}_q$ are the coordinates of a target and a source patch respectively, centered at pixel p and q respectively. The term E_{col} is an extended patch difference measure which we explain in the following section, and E_{3D} imposes multi-view coherence (section 3.7).

3.4.3 Algorithm

We minimize the energy in two steps: initialization, followed by iterative coarse-to-fine PatchMatch and voting. We alternate two *Expectation Maximization* steps for multi-view inpainting, PatchMatch and voting similar to Wexler et al. [129].

For the initialization, we first use 3D information to reproject other images into the current view (section 3.5) and then perform a coarse initialization for the remaining unfilled pixels in a multi-view coherent manner (section 3.6).

The first term is the squared distance of the reprojected image $\mathcal{I}_r$ with the inpainting results $\mathcal{I}_i$. The second term is the measure of similarity for the patches :

$$E_{\text{col}}(\mathbf{t}_j, \mathbf{s}_j) = D\left(W(\mathbf{t}_j), W(\mathbf{s}_j)\right) \tag{3.2}$$

where $W(\mathbf{t}_j)$ is the $N \times N$ patch centered at a pixel j and $W(\mathbf{s}_j)$ its associated nearest neighbor. The distance D is the sum of squared differences (SSD) of an eight-dimensional vector using the RGB values of $W(s_j)$ and $W(t_j)$, the normals, and the gradient-based texture features of [86]. Normals and texture features are scaled using $\lambda_N = 0.5$, and $\lambda_{TF} = 0.75$ respectively.

We use several images of the multi-view dataset as sources for matching and present an algorithm to enforce multi-view consistency during initialization and voting. We encourage multi-view consistency through the term E_{3D}, described in section 3.7, and thanks to a careful initialization which we describe in section 3.6.

Since the reprojection term and the multi-view patch synthesis terms are quadratic, the optimal solution is a linear blend of the patch synthesis result and I_r with alpha as the mixing coefficient. Specifically our algorithm blends the inpainted image $\mathcal{I}_i$ with the reprojected image $\mathcal{I}_r$ using the α weight for iteration t:

$$\mathcal{I}_i \leftarrow \alpha(\mathcal{I}_r) + (1 - \alpha)\mathcal{I}_i \tag{3.3}$$

The new image $\mathcal{I}_i$ is then used in the next iteration of randomized PatchMatch, ensuring that the first term and the overall function $E(\mathcal{I}_i|\mathcal{M}_i)$ are minimized.

We summarize our approach in algorithm 1. It is important to note that to achieve multi-view consistency all steps are done together for all the images being inpainted.

3.5 Reprojection initialization

We use image-based rendering (IBR) to reproject from different images of the input dataset to the target image $\mathcal{I}$. Methods using oversegmentation provide high-quality results [139, 21, 90]; however they are not specifically designed for *inpainting*. Such methods often assume that the missing data for a novel view is in the nearby input cameras, which is not always the case in our context. We reproject several other input images which provides pixels in M_i, thus completing the empty region as much as possible (Figure 3.2, right). The quality of the reprojection degrades rapidly with distance between

Algorithm 1: Global inpainting algorithm

Input: Multiview dataset with 3D reconstruction and binary masks
Result: Inpainted multiview dataset
for *each image* $\mathcal{I}_i$ *and mask* $\mathcal{M}_i$ *in the dataset* **do**
  for *each other image* $j \neq i$ *in the dataset* **do**
    Re-project into view $\mathcal{I}_i$ for pixels $\in \mathcal{M}_i$;
  $\mathcal{I}_i^r$ = min-cut over reprojections $\mathcal{M}_i$;
for *All images* $\mathcal{I}_i$ *in the dataset* **do**
  Initialize colors, normals, depth and Nearest-Neighbor Field (NNF) at coarsest scale ;
for *ScaleResolution = coarsest* **to** *finest* **do**
  repeat
    for *All images* $\mathcal{I}_i$ *in the dataset* **do**
      Reconstruct image $\mathcal{I}_i$ from NNFs with multi-view coherent voting;
      Blend image $\mathcal{I}_i$ with reprojection $\mathcal{I}_i^r$;
    for *All images* $\mathcal{I}_i$ *in the dataset* **do**
      Find NNFs with Patchmatch ;
  until *convergence*;
  if *ScaleResolution* $\neq$ *finest* **then**
    Upscale NNF ;
  else
    for *All images* $\mathcal{I}_i$ *in the dataset* **do**
      Reconstruct image $\mathcal{I}_i$ from NNFs and blend with reprojection $\mathcal{I}_i^r$;

cameras, creating a tradeoff between missing content and low quality reprojection depending on the number of images reprojected.

3.5.1 Reprojection with graphcut

Naive reconstruction of such reprojections (such as median or mean) does not provide satisfactory results, and IBR blending methods [14, 21] typically sacrifice quality for speed. We propose a solution based on a Markov Random Field (MRF) to choose between the input images, by considering *the pixels of* each reprojected image as a label, similar to Whyte et al. [130].

We have a label ℓ for each possible source image (camera) and we seek a label ℓ_p for each pixel p. We first create a median image of all reprojections, and following the work

of Whyte et al. [130], we set unary $V_1(p, \ell_p)$ for a pixel p associated to a label ℓ_p as the squared difference to the median. We ignore pixels belonging to the mask for any input image, by setting $V_1(p, \ell_p) = +\infty$. Any target pixel with all labels equal $+\infty$ will be completed later by inpainting.

Images reprojected from other views do not always provide reliable information, notably when taken in directions far from the target view. We decrease the weight of a reprojected view with the angle between cameras as follows:

$$V_2(p, \ell_p) = \left(|\gamma_{\ell_p,\mathcal{I}}| + 1\right)^2 - 1 \tag{3.4}$$

where $\gamma_{\ell_p,j}$ is the angle between the cameras ℓ_p and j. The energy minimum is zero when the cameras align, and increases with the camera separation. We use a decreasing function of camera angle to allow almost linear falloff when approaching zero, and include a gradient term as described by Whyte et al. [130], to preserve image structure:

$$\begin{aligned} W_1(p, q, \ell_p, \ell_q) = &\|I_{\ell_p}(p) - I_{\ell_p}(q)\| + \|I_{\ell_q}(p) - I_{\ell_q}(q)\| \\ &+ \lambda(\|\nabla I_{\ell_p}(p) - \nabla I_{\ell_p}(q)\| \\ &+ \|\nabla I_{\ell_q}(p) - \nabla I_{\ell_q}(q)\|) \end{aligned} \tag{3.5}$$

where $\nabla I_j(p)$ is the gradient in RGB space associated to pixel p in image j. The colors and gradient have bidirectional terms so they match in the two cameras ℓ_p and ℓ_q, and weight λ is a positive parameter, set to 10 in all our examples.

The global cost function we minimize with the MRF is thus (dropping dependence on ℓ_p for clarity):

$$E(L) = \sum_{p \in P} (V_1(p) + V_2(p)) + \sum_{(p,q) \in N} W_1(p, q) \tag{3.6}$$

where L is the labeling of pixels to inpaint P. The region to inpaint is expanded by a few pixels using dilation to achieve a coherent visual transition between the target region and the rest of the image. N is the set of all pairs of neighboring pixels in P. The α value used in Equation 3.1 is the residual energy value of the min-cut algorithm for each pixel, since it is based on the separation between cameras, which is a strong indication of high quality reprojection.

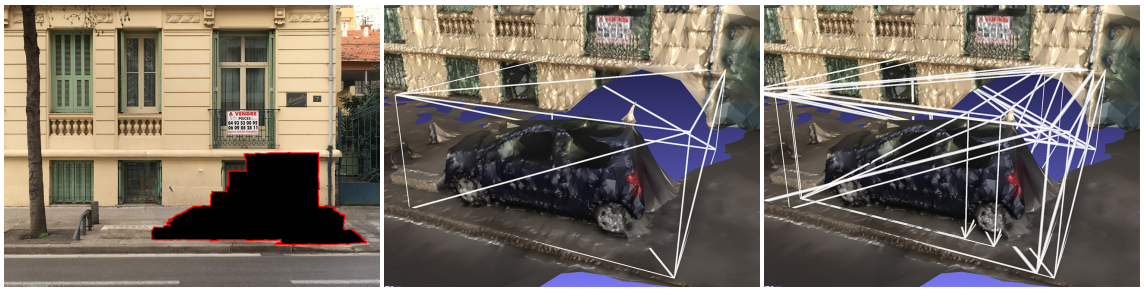


Figure 3.3: **From 2D to 3D boundaries**. Left: 2D boundary of $\mathcal{M}_i^r$ in image i in red. Middle: Corresponding 3D bounding box. Right: Several bounding boxes combined together representing the car object.

Color harmonization. Since the photographs come from a multi-view dataset, the same object observed from different viewpoints may appear with different colors, due to slight non-diffuse materials or subtle changes in lighting between shots. To avoid these artifacts, we use Poisson image compositing [27] after the mincut.

3.6 Coarse initialization

Given the reprojected images $\mathcal{I}_r$, we need to initialize the color, depth and normal values for the remaining unfilled pixels, e.g., black pixels, see Figure 3.2(right). This step is critical for the overall success of the inpainting process. Sophisticated methods such as the Onion Peel approach [86] produce plausible results, but hinder multi-view coherence since they "push" the solution to different local minima for each input image. Instead, we interpolate information from the valid boundary pixels of the masks (i.e., containing information from the original image or reprojection) in a scanline fashion, described in section 3.6, which works well for the street-view scenarios we consider here. We discuss a possible generalization in section 3.9.

Multi-view coherent initialization. In many cases, masks of different objects overlap in some views (e.g., a slanted view of a line of cars in a street). Interpolating over the entire merged mask tends to overblur the result. To avoid this, we introduce the notion of "object being removed", restricting the effect of interpolation to semantically similar regions. This is natural in our context, since masks are either automatically extracted, in which case they correspond to an object class ("car", "pedestrian" etc.) or are created by our user-assisted approach, in which case objects are typically being removed (see

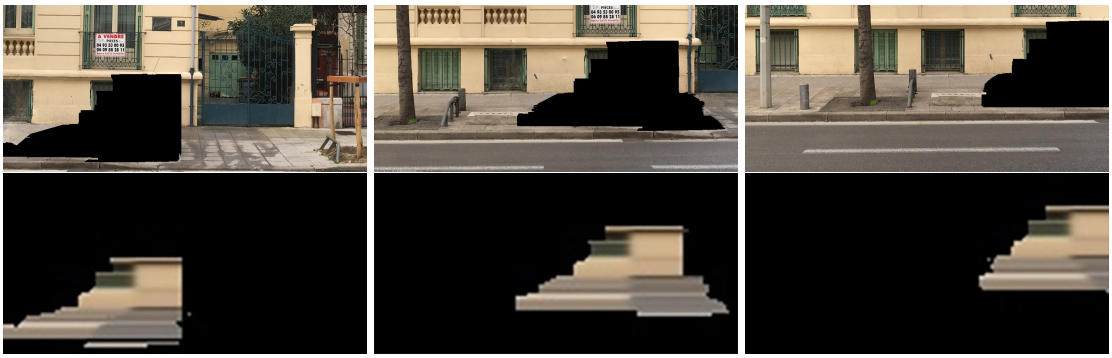


Figure 3.4: **Multi-view coherent initialization**. Top view, $\mathcal{I}_r$ after reprojection. Lower row: intialization with multi view coherence. The reference view is in the middle. The different features are coherent across the views.

section 3.8.1).

To define objects, we take a 2D boundary of each 2D connected component of the mask in each image (shown in red Figure 3.3, left). Then wecreate the 3D bounding box (white in the figure) of the corresponding 3D points contained within the 2D boundary (i.e., reprojected points that have depth). For each pair of 3D bounding boxes A and B we compute $w_{AB} = \max\left(\frac{V(A\cap B)}{V(A)}, \frac{V(A\cap B)}{V(B)}\right)$ and connect them if $w_{AB} > w_{\text{limit}}$ where $w_{\text{limit}} = 0.6$. These connected components are the objects subsequently used in initialization. For each scanline associated to the 2D boundary (Figure 3.3, left) we find the list of pixels on the left and right sides which have color, normals and depth available. For each scanline we linearly interpolate color, normal and 3D points between the two endpoints if their depths are available. If one endpoint does not have reliable depth, we propagate the color and normal of the other endpoint to the border of the image. Depth is considered reliable if it comes from the reconstructed mesh rather than the depth propagation step (subsection 3.4.1). For 3D, we use the plane defined at the existing endpoint by its normal and propagate this across the scanline.

If we initialize each image separately, the result can be very different in each view, no matter what the initialization method used. To enforce multi-view coherence, we choose a reference view which will serve as a guide for all other views of the same object. For each object, we find a reference view by selecting the view with the largest number of scanlines with both left and right samples available, providing the largest amount of information. We reproject the lines of the reference view into all other views containing

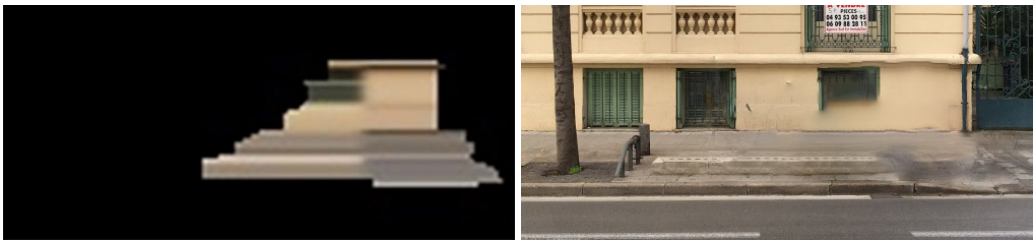


Figure 3.5: **Vertical structure heuristic** Left: Initialization Right: Final inpainted result. A window-like structure is created thanks to the the sigmoid heuristic.

the object and for each pixel of a reference scanline. We copy the color and normal values into the target image and replace the initialization values. For depth, we use the correponding 3D point from the reference and reproject it into the current view. The effect of multi-view coherent initialization is shown in Figure 3.4.

Vertical structure heuristic Excessive blurring may occur when two very different colors are found on each side of the interpolation scanline, since inpainting will be unable to find a good match. For urban street-side scenes, we introduce a heuristic to favor vertical structures which often appear on facades (e.g., windows, doors etc). Instead of linear interpolation we use a sigmoid function which creates a discontinuity which subsequent inpainting transforms into structure, see Figure 3.5. Specifically, we interpolate according to a weight on a sigmoid function across the line, favoring the creation of discontinuity, which subsequent inpainting steps will typically transform into a vertical structure. Specifcally, we interpolate with the following weight $w = \frac{1}{1+e^{\lambda t}}$ where $\lambda = 50$ and $t = \frac{p_x - p_{\bar{x_c}}}{p_{\bar{x_r}} - p_{\bar{x_l}}}$; $\bar{x_c}$ is mean of the centers of the scanlines, and $\bar{x_l}$ and $\bar{x_r}$ the means of the left and right points respectively. The effect of this heuristic is shown in Figure 3.5(b), allowing the creation of a window-like structure.

3.7 Multi-view coherent inpainting

Given multi-view coherent initialization, we can now proceed with our multi-view inpainting to minimize Equation 3.1. Both the PatchMatch search step and the voting of algorithm 1 will use multi-view information. It is important to balance the tradeoff of imposing multi-view coherence and to avoid blurring which can happen due to slight inaccuracies in the reprojection and the geometry used. Thus we first need to define the neighbors of a given image to ensure multi-view coherence.

3.7.1 Defining multi-view coherent neighbors

In street-side datasets, a given object is typically viewed by several input cameras. A simple nearest neighbor approach to define neighbors is insufficient since, if we consider a 2-neighborhood graph, disjoint graph cliques might be formed. Instead we use a minimum spanning tree approach on a graph connecting input views sharing objects with a low-weight edge or a path of low-weight edges.

Each node of the graph corresponds to an input view, with an edge between each pair of views. Consider the pair of views i and j; we intersect all bounding boxes of connected regions to be inpainted (see section 3.6), and the weight w_{ij} is:

$$w_{ij} = \frac{d_{ij}}{\sum\limits_{(A,B)\in B_{ij}} \mathrm{V}(A \cap B)} \tag{3.7}$$

where d_{ij} is the distance between the two cameras i and j, and B_{ij} is the set of pairs of 3D bounding boxes induced by the camera i and j and V is the volume. The idea is to enforce the consistency between views that are close to each other and that share the maximum inpainted content.

Then we find the minimum spanning tree of the graph. We traverse this tree from each node in order of smallest edge weight until we have K neighbors for each node. We use $K = 2$ for computational efficiency unless stated otherwise.

3.7.2 Multi-view search and coherent voting

For a given image i, the search step to create the NNF in algorithm 1 uses the image itself and its K neigbhors. The nearest neighbor of a patch is defined as the closest match amongst the matches in the three images.

Multi-view voting is expressed as the term E_{3D} in Equation 3.1 and is given as follows:

$$E_{3D}(p) = \sum_{q\in S_p} ||I_{c_p}(p) - I_{c_q}(q)||^2 \tag{3.8}$$

where pixel p is from camera c_p and S_p is the set of pixels q from cameras c_q such that q belongs to the mask of c_q and reprojects into p. This reprojection is performed using the current depth estimation inside the hole.

We synthesize new 3D points which correspond to inpainted pixels with color in each view. The E_{3D} term encourages the newly inpainted pixels corresponding to the same 3D point to have the same color.

We use the coarse 3D reconstruction and the inpainted depth to achieve multi-view coherence for voting. For a given pixel i from camera c_i, we look for patches that overlap this pixel and are centered at t_j, and their associated nearest neigbhor in the source s_j gives us a list of color candidates.

For multi-view consistency, we reproject the pixel in its neighboring views c_j. Then we also find patches in view c_j that overlap the reprojected pixel and add the color associated to their nearest neighbor to the list of candidates. The final color is obtained by filtering all these candidates with the following weights:

$$w_j = e^{-\frac{||D(t_j,s_j)||^2}{2\sigma_i^2}} \times e^{-s||d_{c_i,c_j}||^2} \times Q(t_j). \tag{3.9}$$

Following Newson et al. [86], the first term favors source patches that are similar to their associated target patches. The parameter σ_i is defined at the 75th percentile of all distances $||D(t_i, s_j)||$. The second term gives more importance to closer views. The parameter s is the number of scales from the coarsest scale to the current scale. The idea is to reduce the influence of the multiview coherence as the upscaling occurs to avoid blurring at the finest scales. The quality term Q is a measure of how much the nearest neighbor field is constant and in the same image. This is inspired by a similar approach previously used to improve single-image inpainting [28]. It is defined for a pixel target t_i as one plus the number of "high quality" neighbors t_j such that $t_i - t_j = s_i - s_j$, where the neighors are taken from a 4-neighborhood around t_i.

Multi-view coherence has a significant effect on the results (Figure 3.6). Without coherence, two different images of the dataset can have very different inpainted content. Using our approach, similar structures are created in the same position, which is central for good-quality IBR.

3.8 Results and comparisons

We tested our results on seven multi-view datasets. Two are previously available [21] (e.g., Figure 3.10 middle set), four were shot using a cellphone camera in a casual capture setting, (Figure 3.10 top, Figure 3.9) and one is from a Google Street View data

Figure 3.6: Above: Result without MV coherence; note missing blocks on the pavement for left image. Below: Result with MV coherence.

(last dataset of Figure 3.10), which is extremely challenging since the baseline between panoramas is very large, causing SfM to often fail.

Processing Google Street View data For Google data, we download the panoramas via the web interface, and extract 3-7 rectified images typically facing the facade and at +/-30 degrees (and sometimes looking up and/or down). We also use images from different dates to improve SfM and reconstruction. The reconstruction algorithm often produces spurious polygons instead of a floor for the Google data. We removed all polygons below ground level manually for these examples and we do not use the normals for the SSD of the term in Equation 3.1 of the main document, since they are unreliable.

3.8.1 Usage scenarios

We show two applications of our approach, one using a semi automatic method to remove specific classes of objects, the other introducing an interactive multi-view editor for IBR.

Semi-automatic object removal. We use the automatic object recognition approach of Gidaris and Komodakis [41], for the classes “car”, “people” and “motorbike”, and we

Figure 3.7: **IBR editing**. Left: input image. Right: novel view with the pillar extracted and moved, revealing content behind it. Our method allows such editing in a free-viewpoint IBR setting.

use the bounding boxes with the highest scores. This method works well in general; we increase the size of the bounding boxes to avoid missed regions and for some datasets, we had to correct manually for missed regions (a few minutes per dataset).

These bounding boxes are used as masks for our multi-view inpainting. We use the inpainted images and the synthesized depth and run SLIC oversegmentation and additional depth synthesis [21] for remaining unreconstructed regions if required. We can now run IBR on the scene with the objects removed.

Editing multi-View captures for IBR. Another application of our approach is to not only remove objects in an IBR scene, but also to be able to move them. This enables limited editing of multi-view captures with free-viewpoint IBR.

To do this, we built an interactive selection tool for multi-view datasets. We use an oversegmentation which allows selection of fine details to create good quality masks. The user can create variable width strokes on the object to select and segment it. We use multi-view stereo patches to reproject the strokes onto segments in the other views. The user can then cycle through views and complete or correct the segmentation, with little effort. We show an example of such a session in the accompanying video [1].

Once the object is segmented, we either use the resulting masks or combine them with masks from the automated approach. We extract the part of the image from the original

[1] http://www-sop.inria.fr/reves/Basilic/2016/TSPD16/

image and render with a two-pass approach: first we render the background, and then render the extracted object, allowing us to edit the IBR scene, as seen in Figure 3.7 and the video.

3.8.2 Comparisons

All comparisons were run by the authors of the previous papers. We compare with two single-image methods: PatchMatch (using content-aware fill in Adobe Photoshop) and the method of Huang et al. [57]. Single image methods have difficulty inpainting large regions, even when attempting to deduce information from planes as in Huang et al. [57]. We compare to two multi-view methods. The method of Whyte et al. [130] produces good results in many regions, but is not multi-view consistent and is not designed for large regions with no reprojection. Even though the graphcut energy is infinity in masked regions, the method sometimes copies information from masked regions in other views or leaves the region black. The method of Baek et al. [6] progressively builds multi-view information by successively visiting images. The results are of comparable visual quality to ours, but multi-view consistency is sometimes lost (Figure 3.9). While depth information can be consistent across neighboring views (Figure 3.8, mid-right), it is not consistent at more distant viewpoints (left). The depth in this method is not globally consistent and lacks detail: e.g., the difference in depth between floor and wall in Figure 3.8(left) is minimal. Given the lack of global depth consistency and detail, this solution cannot be used for IBR. While our synthesized depth is not perfect, it is sufficient for use with IBR as seen in the video [2].

3.9 Conclusions and discussion

Our results still suffer from some residual bluriness, that is hard to remove without breaking multi-view coherence, given the inaccuracies of the 3D reconstruction. Initialization could be improved using the 3D reconstruction to propagate structures in any direction, instead of the current horizontal interpolation on scanlines. Analyzing the 2D boundary of the reprojected region to identify directional structures could also improve results.

We demonstrated a multi-view consistent inpainting algorithm which enables editing of

[2] http://www-sop.inria.fr/reves/Basilic/2016/TSPD16/

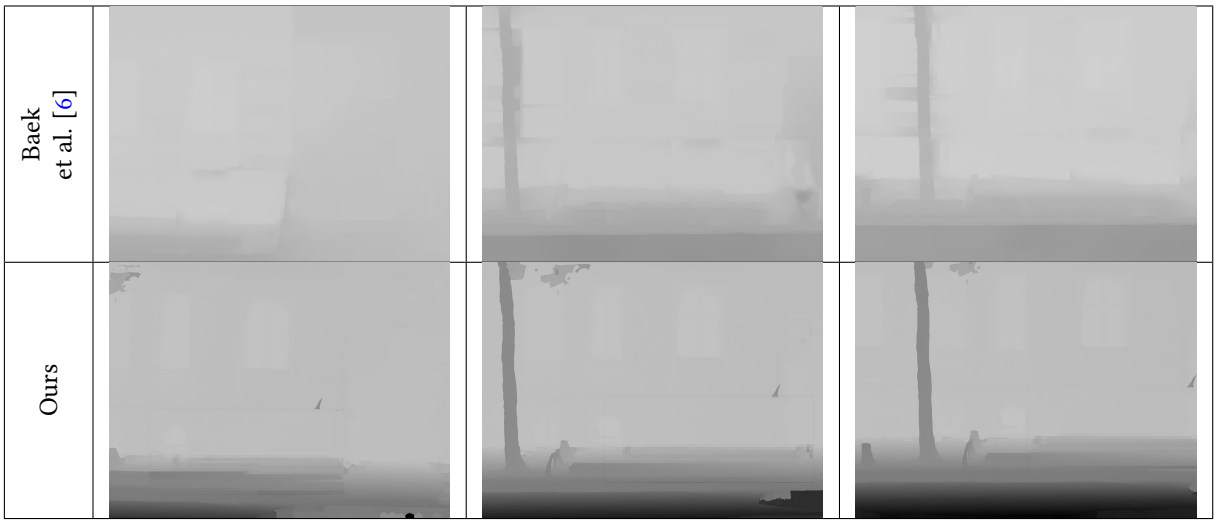


Figure 3.8: Comparison of depth of Baek et al. [6]; note incorrect depth on pavement (top left). Lack of global consistency and depth detail renders this method unsuitable for IBR.

IBR scenes in a free-viewpoint setting. By extending single image patch-based inpainting to a wide-baseline multi-view context, we allow the removal of objects from an IBR scene. These removals allow increase in IBR rendering quality when applied to objects which are difficult to reconstruct via MVS such as cars or moving pedestrians. They also allow to move, anywhere in the scene, objects that are well reconstructed, providing a first step into the editability.

This work was partially funded by the CR-PLAY[3] European project, aiming to reduce the difficulty of content creation for video games. As IBR methods provide a cheap and easy way to capture and render real environments, they are a suitable framework for the project goals. Our work fits this objective well as game artists and designers want to have control over the games assets.

Our method has also led to the development of a follow-up paper by Philip and Drettakis [97], extending the robustness and scalability of our method by propagating 2D structures and performing the inpainting directly into those structures.

[3] http://www.cr-play.eu/

Figure 3.9: **Comparison with other methods**. We achieve much better image quality than all single-image inpainting techniques (rows 1-3). Compared to the method of Baek et al. [6] which is multi-view, we see in the red boxes that our result has better multi-view consistency.

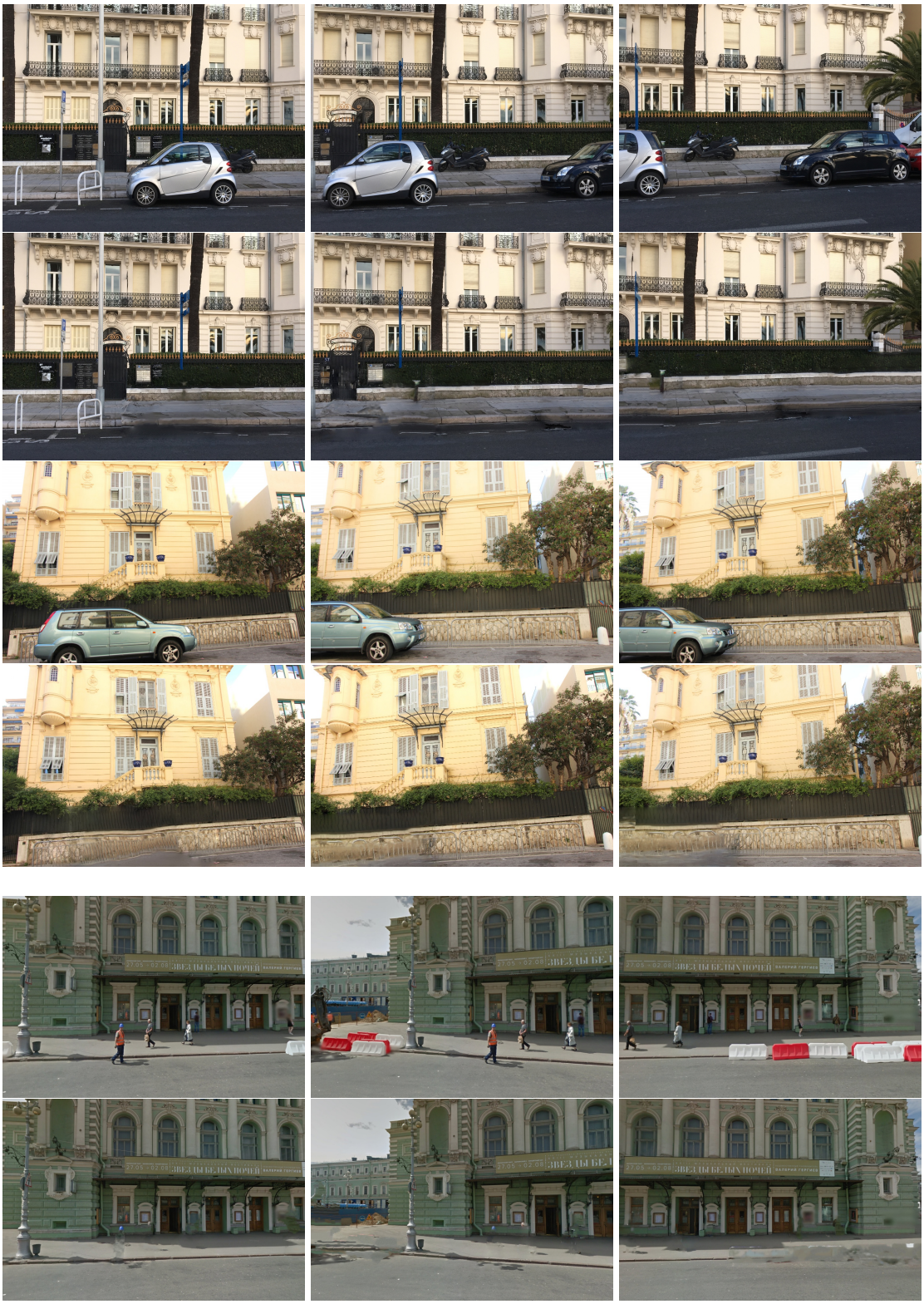


Figure 3.10: First 2 rows: results from one of our datasets. Middle 2 rows: the Yellowhouse datasets from Chaurasia et al. [21]. Lower 2 rows: dataset reconstructed from Google StreetView. Input image above and and the resulting inpainting below for all cases.

Figure 3.11: Results for 4 datasets. Each pair of rows shows the input image (above) and the resulting inpainting (below). In the toy dataset, a barrel and one character have been removed using the interactive tool.

Figure 3.12: Results for the Street-10 and Yellowhouse-12 datasets from Chaurasia et al. [21].

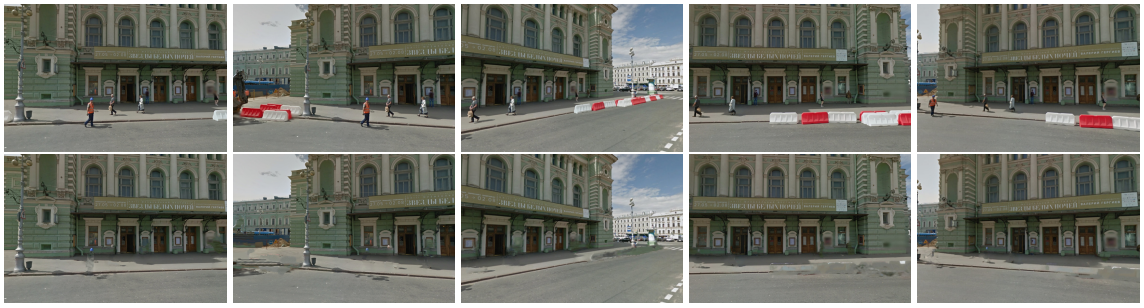


Figure 3.13: Results for a scene in St. Petersburg captured from Google StreetView. These datasets are particularly challenging as described in paragraph section 3.8.

Chapter 4

Reconstructing thin structures

Contents

4.1 Introduction

We have seen that recent IBR algorithms [64, 74, 90, 95] provide a compelling solution to virtual visits and photo tourism, avoiding the expense of 3D modeling/texturing and complex photo-realistic rendering. Their key advantage is the simplicity of the input: only a set of photos of a scene is needed, yet they allow high-quality free-viewpoint rendering.

However, one of the key problems in IBR methods is the rendering of regions where 3D reconstruction is hard or impossible, such as vegetation, reflections and *thin structures*. In the previous chapter, we have seen that the removal of such problematic regions allows to widen the scope of IBR scenes. Still, we de not consider the case where we want to actually render those difficult type of objects.

Previous methods have addressed reflections and transparency [112, 64], or vegetation [21] but no solution currently exists for thin structures. These are present in all man-made environments: outdoors, fences or stair banisters are very common, while indoors a variety of everyday objects and utensils have similar properties, e.g., grills, racks or decorative elements. In this chapter, we focus on thin structures that are supported by user provided simple geometries, such as planes or cylinders.

We take as input a set of photographs of the scene from multiple viewpoints as well as a traditional 3D reconstruction from off-the-shelf structure-from-motion (SfM) and multi-view stereo (MVS) methods. The 3D reconstruction is usually incorrect for the thin structure; our goal is to segment it out on the correct 3D supporting geometry to improve image based rendering. Multiple factors make this problem hard. The structures are very thin and often lack texture; as a result standard descriptors are ineffective and regularization is difficult. Also, the see-through nature of these objects makes multi-view inference challenging.

Our problem is related to *de-fencing* methods (e.g., [94]) but their objective is often limited to removing occluding fences, in which case segmentation can be conservative and less precise. The foreground layer can be more precisely estimated [134], but this requires a small baseline in video sequences.

The key intuition in our work is that we can use multi-view information and the partial 3D reconstruction to estimate segmentation of thin structures in multiple input views.

In addition to the images and reconstruction we use as input, a short user interaction step is needed to define the supporting geometry and region of interest. The first part of our algorithm is a Markov Random Field (MRF)-based multi-view segmentation, which can handle *multiple layers* of thin structures. We combine appearance cues from color models, median colors and variance, with multi-view consistency constraints on segmentation results. The second part of our solution is an IBR algorithm that allows *free-viewpoint* rendering of multi-layer thin structures. For a given fragment, our renderer interprets alpha values as probabilities to be on a thin structure. These are used to blend weighted colors from the different views.

Our contributions can be summarized as follows:

- A multi-view segmentation algorithm for thin structures supported by simple geometries, which uses multi-view links and color variance to resolve ambiguous cases.
- An end-to-end solution to IBR for such *multi-layer* thin structures, including pre-processing to ensure accurate segmentation, post-processing to generate a clean background, and a new IBR algorithm that allows *free-viewpoint* navigation of scenes containing these structures.

Our results show significant improvement over previous work in the identification of the thin structures, in the resulting segmentation, and in the quality of the image-based rendering in free-viewpoint navigation.

4.2 Related work

Our work is related to image *de-fencing*, repetitive structure detection and multi-view segmentation. We also briefly discuss aspects of 3D reconstruction and Image-Based Rendering research related to our work.

4.2.1 De-fencing and repetitive structure detection

Hays et al. [48] divide research on discovering repeated elements into two extremes: the first focusing on the individual element [79] and the other imposing strong structure priors on the general layout of the repeating elements [125]. Hays et al. are the first to automate lattice detection in real images without pre-segmentation. Further improvement is

proposed by Park et al. [93] by solving the problem in an MRF setting. More recently, Liu et al. [77] avoid using interest points and apply the Generalized PatchMath algorithm in combination with Particle Belief Propagation to infer the lattice structure. In the case of facades, vanishing lines can be used for plane detection and rectification [133]. In this case dense descriptors are matched not only using repetition but also symmetry. In a multi-view setting, it is possible to detect repetitive elements on more complex surfaces using reconstructed 3D geometry and the images [59]. This however cannot be applied to thin structures in our scenes as there is often no reliable 3D reconstruction.

The problem of image *de-fencing* consists of removing fences from pictures or videos. Liu et al. [78] were the first to propose an automatic method to detect and segment fences in images. Texture based inpainting [26] is used to fill the extracted regions. Fences are found by searching for a lattice that explains the relationship between repeated elements in the image [48]. This method is further improved by Park et al. [94] using a multi-view approach for inpainting and a different lattice detection algorithm [93]. In the case of videos, optical flow is the main cue used to identify fences. A first method is proposed by Mu et al. [84] based on motion parallax. Recently, a robust method for obstruction free photography was proposed by Xue et al. [134] to handle occlusion from both fences and windows, which generates an alpha matted thin structure layer. Yi et al. [136] also rely on motion in their fence/non fence segmentation. Finally, Yamashita et al. [135] use multi-focus flash/non flash images to identify regions corresponding to objects closer to the camera. Most of these methods assume that the fence is closer to the camera than our typical input. In video-based methods, optical flow estimation is central, and often fails on the wide-baseline input data we have. We show several comparisons to previous methods in section 4.7 which demonstrate that these methods are not adapted to our goals.

4.2.2 Multi-view segmentation

Our work is also related to multi-view segmentation methods that try to identify the foreground object visible in different viewpoints. Some of these methods rely on consistency of the projection of 3D points [62, 32] which is unreliable with thin structures. Other segmentation approaches use constraints from stereo [65] or matching along epipolar lines [16]. Both strategies are not designed to handle ambiguities due to the often repetitive pattern. Wexler et al. [128] estimate two layers from multiple images using a Bayesian

framework. Contrary to our work, they assume foreground and background transformations are modeled by planar projections. The resulting maximum-a-posteriori estimation requires the definition of a reference view which is not trivial when considering more complex camera setups. Our approach is able to handle complex 3D geometries and multiple layers. Thanks to our multi-view constraints, segmentation is estimated in the original input images for best rendering results.

4.2.3 Multi-view stereo reconstruction and IBR

Thin structures are also an important limitation for multi-view stereo (MVS) reconstruction methods [43, 39, 58]. Ummenhofer and Brox [126] propose a method to reconstruct thin objects which have almost no volume compared to the surface size. Because of the errors in the 3D reconstruction and the normals, the object is not reconstructed (e.g., a sign observed from opposing viewpoints). In a different setup, Oswald et al. [92] enforce connectivity constraints and surface genus in temporal 3D reconstruction. In their paper on scene abstraction, Hofer et al. [55] use 3D lines to represent 3D scenes. There also has been work specific to the reconstruction of wire structures from images [76, 82, 18]; these methods have strong assumptions about the simplicity and/or tubular structure of the objects being reconstructed, and thus do not apply to our context. In general, the hypotheses on input in all of these methods are very different from ours, making them inappropriate for our data.

We have seen in chapter 2 that recent IBR algorithms use MVS reconstruction to provide high-quality free-viewpoint navigation when reconstruction works well. The harder case of reflections has been addressed with explicit reflection reconstruction [112], gradient domain rendering [64] or stock 3D models [91], while vegetation can be handled using oversegmentation and depth synthesis [21]. Layered Depth Images [51] can be used with image data based on an ordering algorithm to allow correct alpha blending; in contrast we interpret alpha values as probabilities to allow specific visibility processing for our multi-layer semi-transparent thin geometries. In recent work, the Soft3D algorithm by Penner and Zhang [95] uses a volumetric depth-sweep approach for IBR with a set of clever filtering steps, based on guided filtering and soft visibility, with excellent results. The volumetric nature of this approach means that very fine resolution at unacceptable storage/computation cost would be needed to represent the thin structures we treat. Our rendering algorithm shares some ideas with the soft visibility approach of

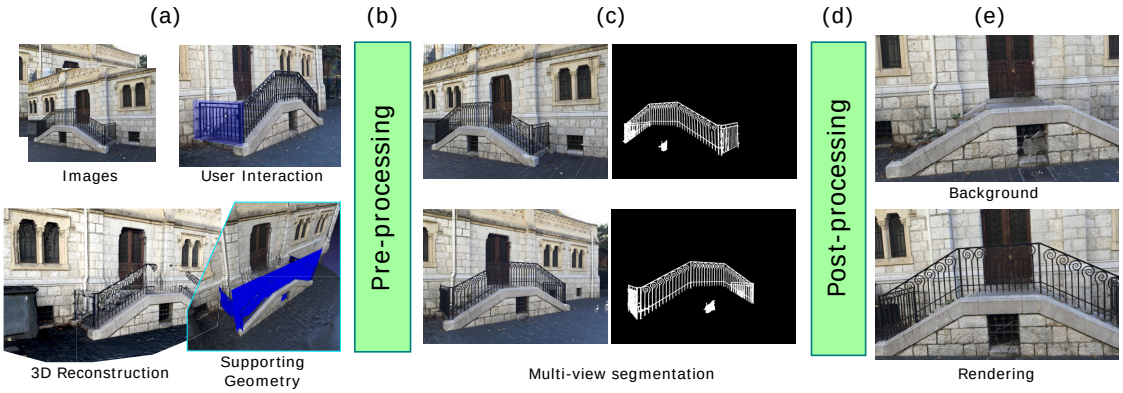


Figure 4.1: **End-to-end solution for thin structures in IBR**. (a) After reconstruction and supporting geometries extraction from user interaction, we proceed to (b) pre-processing step to remove spurious geometry. Then we perform (c) multi-view, *multi-layer* segmentation followed by (d) post-processing to create "clean" background images and geometry. The result can be used by our multi-layer rendering algorithm (e), allowing *free-viewpoint* navigation.

Soft3D which we discuss in more detail when presenting our method. The central difference is that we work with surface-based 3D (MVS reconstruction and thin supporting geometries) rather than volumes, which allows better visual quality for free-viewpoint navigation far from the input cameras.

4.3 Overview

Our pipeline is shown in Figure 4.1. We start with a set of photographs of the scene together with a 3D reconstruction, estimated using multi-view stereo (e.g., [58, 101]): We call this the *proxy* geometry and we assume that it completely covers the background of the scene. The 3D model usually represents the thin structure poorly and we provide a user interface to specify the supporting 3D geometry of the thin structure. The user manually specifies 3D points in images, by clicking in 2 images per point. For a plane, 3-4 points are needed and for a cylindrical segment, the user specifies 4-5 points for each of the "upper" and "lower" circles defining the cylindrical segment. The output is a collection of meshes that roughly cover the thin structures (illustrated in Figure 4.1.a). This user interaction takes no more than a few minutes for all our examples.

Our goal is to segment pixels belonging to the thin structure in all the input views (Figure 4.1.c). There can be multiple overlapping layers of see-through thin structures. As a result, in addition to separating the thin structures from the background, we also need to

segment the thin structures into a distinct set of layers. This makes traditional appearance terms less reliable. Moreover, the segmentation of such structures is hard because the regions are not compact and challenge the balance between area and contour terms in traditional segmentation. Our key observation is that we can leverage the multiple input views to resolve ambiguity present in a given view. Consider Figure 4.2: the structures on the left are very hard to extract as they have colors very similar to the door behind. However, the same structures in the right image have high contrast and can be extracted easily.

Our segmentation algorithm relies on color models and multi-view constraints: for occluded layers, we use median colors [128] to leverage the multiple views and obtain a color model that is more robust to occlusions. However, for the front-most layer, any inconsistency in colors between the different views is likely to indicate incorrect segmentation and using color variance across viewpoints is more effective. In addition to this, we use multi-view links to help resolve ambiguous cases for segmentation. The segmentation operates layer by layer, from front to back; we will refer to "foreground" as the current front-most thin structure and "background" as the layers behind together with the non-thin parts of the scene. Because we leverage the discrepancy between reprojections to background and foreground layers, we assume that the thin structure has not been reconstructed in the 3D proxy. To ensure this, we remove geometry in the close neighborhood of the supporting geometry during a preprocess step.

After multi-view segmentation, we refine the segmentation using off-the-shelf alpha

Figure 4.2: **Benefits of a multi-view approach.** Segmentation of the ambiguous area on the left can be resolved using the view on the right.

matting [49], whereas the background regions occluded by the thin structures are filled using the median images and inpainting. The resulting images are then used to create a "thin-structure-free" 3D reconstruction of the background. These correspond to the post-processing in Figure 4.1.d. Details for pre- and post-processing are provided in section 4.5.

We introduce a new rendering algorithm that handles multi-view alpha mattes, by interpreting alpha values as conditional probabilities that a fragment contains a thin structure. We estimate the overall alpha value using Bayes rule. Rendering involves a depth-peeling algorithm on the supporting geometries of the thin structures, rendered after a first Unstructured Lumigraph Rendering [14] (ULR) pass of the clean background (Figure 5.1.e). Our results on scenes with thin structures show significantly better quality than previous methods, especially for the free-viewpoint navigation far from the input cameras.

4.4 Multi-view segmentation

To resolve the difficult multi-layer segmentation problem, we use multi-view color and geometry information.

4.4.1 Multi-layer segmentation

Our approach handles multiple layers of thin structures, e.g, the corner of the staircase in Figure 4.1. For each pixel p in each input image, we create a sorted list of N_p front-to-back depth candidates d_k by ray-casting the proxy and the thin structure supporting geometries. The last depth candidate d_{N_p} per pixel is always the proxy depth because we assume that the proxy is opaque, so we stop ray-casting as soon as it is hit.

We denote by $\mathcal{P}$ the set of all pixels p for which $N_p \geq 2$. It corresponds to all pixels that need to be segmented because they have at least one depth candidate in addition to the proxy depth. Solving the general multi-layer segmentation problem is equivalent to assigning the correct depth to each pixel, which we model as finding a labeling that will minimize an energy function defined by multi-view information on color and geometry.

To solve the multi-label problem, we decompose it into a sequence of binary label problems, in an alpha-expansion-like fashion [13], considering depth labels from front-to-back. For each iteration k, we select all the pixels with current segmentation $\geq k$, and

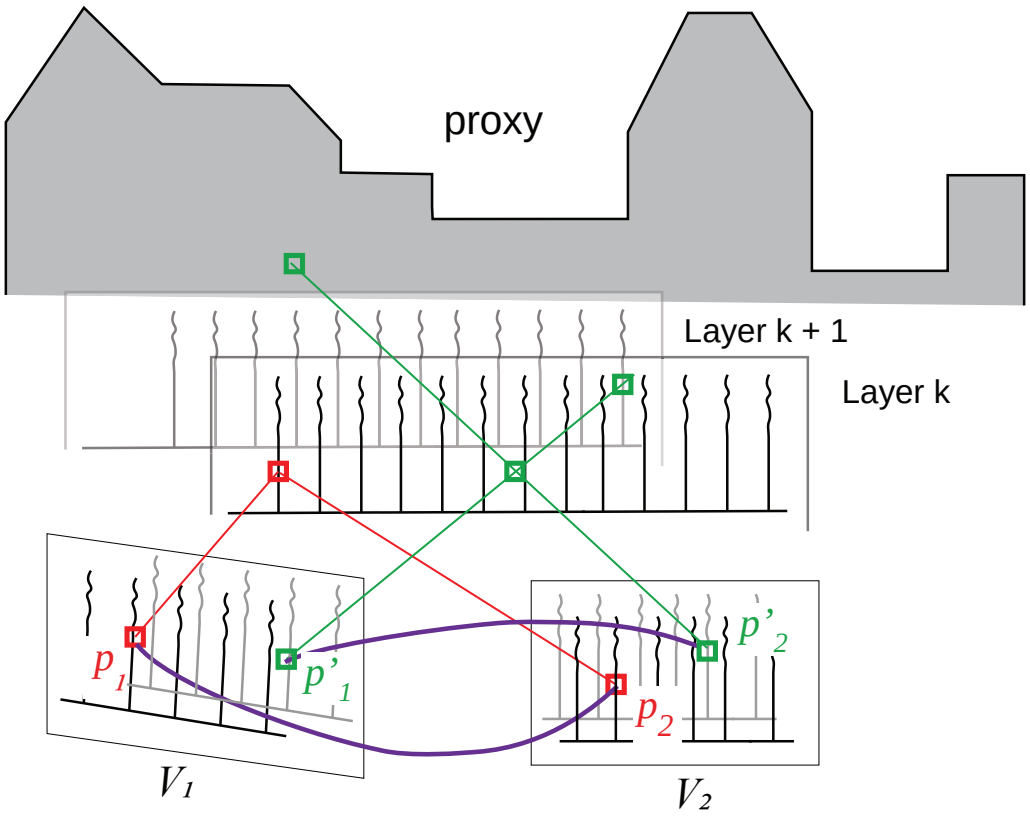


Figure 4.3: **Multi-view links at iteration k.** We create multi-view links by connecting two pixels which correspond to the same 3D point on the layer k (p_1 linked to p_2 and p'_1 linked to p'_2). When the 3D point really corresponds to a thin structure ($p_1 \leftrightarrow p_2$), we have linked together two pixels from the layer k. When the 3D point is in-between the thin structure ($p'_1 \leftrightarrow p'_2$), we do not know which layer corresponds to which pixel: p'_1 is in layer $k+1$, while p'_2 is on the proxy but we can infer that the two pixels are on layers $> k$.

we want to find out how much we can expand the depth layer k. The two labels for the sub binary problems are "depth layer is k" and "depth layer is $> \ k$", i.e., "foreground" and "background" respectively.

To see the justification for this decomposition, consider Figure 4.3: labels of pixels associated through the current depth layer k are linked. They are either both at the correct depth layer k, which is the case for pixels p_1 and p_2, or both at a depth layer $> k$ in the case of pixels p'_1 and p'_2. In this case, it is not possible to associate a specific layer $> k$ to the pixels p'_1 and p'_2 (see Figure 4.3).

We denote $\mathcal{K}$ the geometry corresponding the depth layer k. We use the term *background*

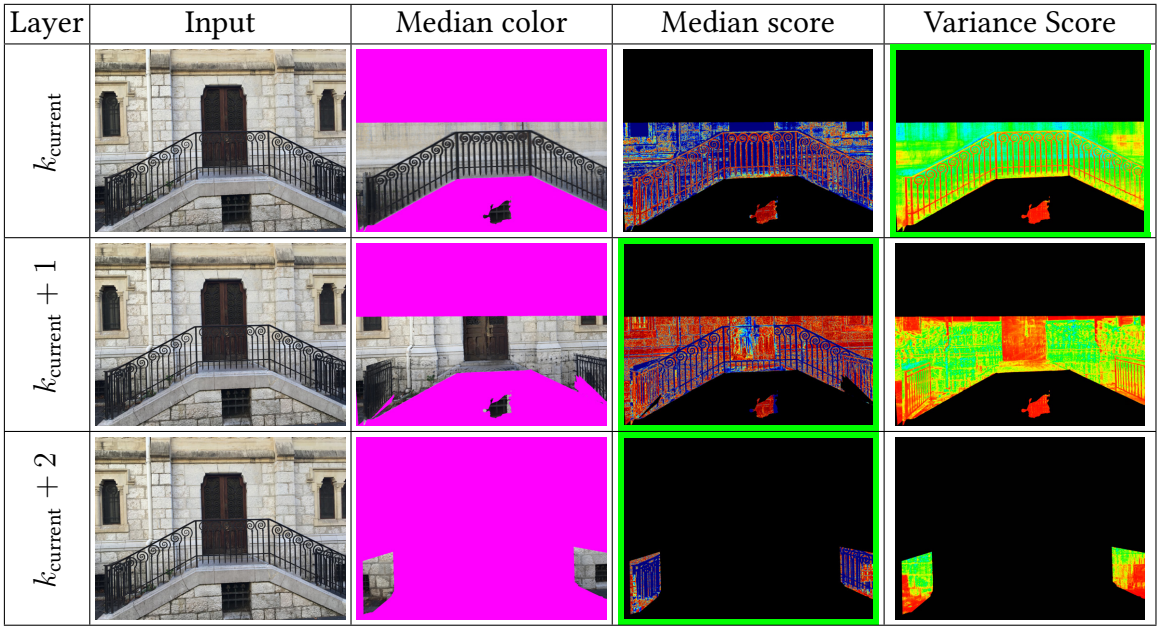


Figure 4.4: **Advantage of using median or variance information for each depth layer for a given current depth layer**. Warm colors in the score columns indicate high likelihood for this layer to give the correct depth. For the current layer, the variances give cleaner information. For the layers beyond, the medians remove outliers while the variance score, more sensitive to occlusions, is low on the wall. For each layer, the most informative score is outlined in green.

geometry to refer to the geometry corresponding to all depth layers $> k$, i.e., thin structures behind the current layer or proxy of the scene. We iteratively solve these binary problems for $k = 1$ to $k = \max_{p \in \mathcal{P}} N_p - 1$.

4.4.2 General formulation

We can now express the segmentation as a binary labeling problem for each pixel, minimizing an MRF energy function. We start with standard color and smoothness terms, and use two terms which exploit multi-view cues. The first provides *a multi-view color prediction* of a foreground vs. background pixel given the reprojections from other views, while the second term *links the label* of pixels from *different viewpoints* if they correspond to the same point on the foreground layer $\mathcal{K}$.

We next describe the graphcut segmentation terms before introducing our new terms in more detail.

Color model : We compute prior probabilities for each pixel for being in the thin structure or in the proxy: $P^{\text{color}}_{\text{thin}}$ and $P^{\text{color}}_{\text{proxy}}$ respectively. These are estimated from global per view appearance models. In our case, we use histograms. We cluster all the colors from all input images. For a color c, we denote its associated cluster $C(c)$, where this cluster defines the corresponding bin in the histogram. We note $\mathcal{P}^{\mathcal{I}}_{\text{thin}}$ the set of pixels of image $\mathcal{I}$ currently segmented as one of the thin structures and $\mathcal{P}^{\mathcal{I}}_{\text{proxy}}$ the set of all input pixels not in $\mathcal{P}^{\mathcal{I}}_{\text{thin}}$. Probabilities for a pixel p from image $\mathcal{I}$ with color c_p are then defined as:

$$\begin{aligned} P^{\text{color}}_{\text{thin}}(p) &= \frac{\#\{q \in \mathcal{P}^{\mathcal{I}}_{\text{thin}}, C(c_q) = C(c_p)\}}{\#\{q \in \mathcal{P}^{\mathcal{I}}_{\text{thin}}\}} \\ P^{\text{color}}_{\text{proxy}}(p) &= \frac{\#\{q \in \mathcal{P}^{\mathcal{I}}_{\text{proxy}}, C(c_q) = C(c_p)\}}{\#\{q \in \mathcal{P}^{\mathcal{I}}_{\text{proxy}}\},} \end{aligned} \tag{4.1}$$

where $\#$ is set cardinality. Finally, the probabilities P^{color}_{k} to be part of the layer k based on the color models are given by:

$$P^{\text{color}}_{k}(p) = \begin{cases} P^{\text{color}}_{\text{proxy}}(p) & \text{if } k = N_p, \\ P^{\text{color}}_{\text{thin}}(p) & \text{otherwise.} \end{cases} \tag{4.2}$$

Smoothness term : We use a standard pairwise contrast sensitive energy term, with a 4-neighbor connectivity. Specifically:

$$E_{\mathcal{N}}(p,q) = \begin{cases} \frac{1}{1+||c_p - c_q||} & \text{if } l_p \neq l_q, \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

with $||c_p - c_q||$ the L_2 norm of the color difference.

4.4.3 Multi-view color term

We leverage multi-view information to determine the probability of being foreground or background by *reprojection*, in the spirit of multi-view stereo [108]. Standard reprojection error approaches are not sufficiently robust in our challenging context, in particular because of occlusion. Instead we compare observed pixels to the *median* of the reprojections from multiple views [128] and introduce *variance* to allow more robust results.

For a 3D point in the current depth layer k_{current}, since we have an approximate segmentation of the previous depth layers, we can compute reliable visibility information

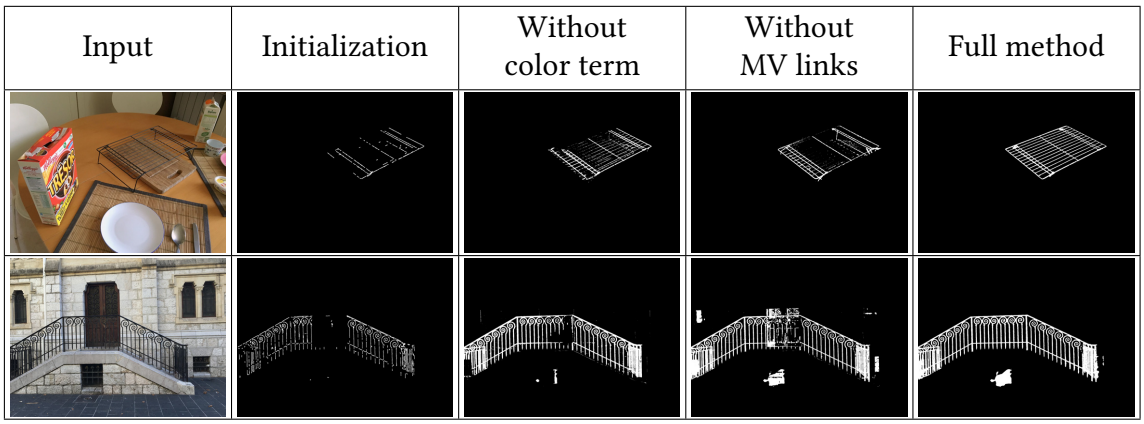


Figure 4.5: **Advantage of using a multi-view approach for segmentation.** The MV links allow us not only to retrieve missing segmentation because of a similar background but also to remove wrong segmentation because of possibly misleading unary terms.

from input views. With the hypothesis of low specularity, a high color variance in the reprojections of the 3D point into the input views is a strong cue for an incorrect depth. On the other hand, for a 3D point in a depth layer $k > k_{\text{current}}$, we do not have reliable visibility information, so even with a correct depth, a high color variance could simply be a consequence of occlusions. However, if we make the assumption that since the foregrounds are thin, a background point will not be occluded in the majority of the views, we can use the median color across views to remove occlusion outliers.

For a given view, we use the 3D information from layer k to project into the other views and collect color samples. The variance map $\sigma_{\mathcal{I}}$ defines the variances of these samples for each pixel p. The median image $\mathcal{M}$ is obtained by sorting the samples by luminance, and keeping the median color value for each pixel.

The likelihood function for a pixel p to be part of depth layer k is

$$\mathcal{L}_k(p) = \begin{cases} \mathcal{G}(\sigma_{\mathcal{I}}(p), 0, \sigma_{\text{ref}}) & \text{if } k = k_{\text{current}}, \\ \mathcal{G}(\mathcal{I}(p), \mathcal{M}_k(p), \mathcal{I}_{\text{ref}}) & \text{if } k > k_{\text{current}}. \end{cases} \tag{4.4}$$

with $\mathcal{G}$ indicating a Gaussian distribution. In the case of the current depth layer k_{current}, a small variance in the color samples is likely to indicate correct depth assignment. In the case of other depth layers $k > k_{\text{current}}$, using the median color map $\mathcal{M}_k$ is more robust to the occasional occlusions. We show examples of the median images in Figure 4.4. In

the case of the front-most layer ($k = k_{\text{current}}$), using the variance map gives more precise results whereas the median colors are more effective on layers further away ($k > k_{\text{current}}$).

Moreover, since the layers occlude each other, we introduce a prior weight λ_k to take into account that the segmentation of layer k becomes more likely as it approaches the front (excluding the proxy which is prominent). Given that the layer $k = N_p$ corresponds to the proxy, we model this by a Bernoulli trial of parameter 0.6 such that $\lambda_{\text{proxy}} = \lambda_{N_p} > \lambda_1 > \lambda_2 > \cdots > \lambda_{N_p-1}$.

Therefore the probabilities P_k^{mv} to be part of the layer k ($k \geq k_{\text{current}}$), based on the multi-view information are given by:

$$P_k^{\text{mv}}(p) = \frac{\lambda_k \cdot \mathcal{L}_k(p)}{\sum\limits_{k' \geq k_{\text{current}}} \lambda_{k'} \cdot \mathcal{L}_{k'}(p)} \tag{4.5}$$

4.4.4 Multi-view links

We introduce multi-view links to enforce consistency between views, transferring more reliable segmentations to views with harder configurations. We reproject each pixel into neighboring images with depth $\mathcal{K}$ and we create a link for each projection if its current segmentation is $\geq k$. We do this since if the reprojection falls on a pixel with current segmentation $< k$, we do not want to change this decision; see illustration in Figure 4.3, To limit issues due to coordinate rounding, links are created only if the backprojection in the opposite direction is on the same pixel.

$E_L(p, q)$ is the energy term corresponding to these multi-view links. It is created between all pairs of views (i, j), linking a pixel p from i with a pixel q from j through reprojection:

$$E_L(p, q) = \begin{cases} 1 & \text{if } l_p \neq l_q, \\ 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

These links allow the propagation of segmentation results from views where color models are more discriminatory to images where the difference between foreground and background appearance is ambiguous.

4.4.5 Unary energy terms

The probabilities described above give information about a layer k. The unary cost associated to a layer k is given by :

$$\mathcal{U}_k(p) = -\log\left(\lambda_{\text{color}} \cdot P_k^{\text{color}}(p) + \lambda_{\text{mv}} \cdot P_k^{\text{mv}}(p)\right) \tag{4.7}$$

Terms λ_{color} and λ_{mv} are mixing coefficients between the global and the multi-view appearance models, where $\lambda_{\text{mv}} = 0.4$ and $\lambda_{\text{color}} = 1 - \lambda_{\text{mv}}$. We now need to compute the unary terms with respect to the labels "foreground" and "background". For the foreground label "k" the unary term is simply the unary cost associated to that depth layer. For the background label "$> k$" the unary term is the minimum of all the unary costs associated to the depth layers $k+1, k+2, \cdots$. We use the minimum so that we are as conservative as possible when assigning the label k, because if k is chosen, the pixel will not appear in the following binary segmentation problems as we solve front-to-back. Therefore the unary energy term is given by:

$$E_U(p) = \begin{cases} \mathcal{U}_k(p) & \text{if } l_p = \text{"foreground"}, \\ \min\limits_{k'>k} \mathcal{U}_{k'}(p) & \text{if } l_p = \text{"background"}. \end{cases} \tag{4.8}$$

4.4.6 Final energy formulation

$E_N(p,q)$ and $E_L(p,q)$ are pairwise terms used to enforce coherent segmentation respectively at image and multi-view level.

If we note by $\mathcal{N}$ all the pairs of neighboring pixels, and by $\mathcal{L}$ all the multi-view links, the final energy optimized by the MRF is as follows:

$$E = \sum_{p\in\mathcal{P}} E_U(p) + \underbrace{\sum_{(p,q)\in\mathcal{N}} E_N(p,q)}_{\text{spatial smoothness}} + \underbrace{\sum_{(p,q)\in\mathcal{L}} E_L(p,q)}_{\text{multi-view smoothness}} \tag{4.9}$$

In Figure 4.5 we illustrate the effect of the unary terms and the multi-view consistency term $E_L(p,q)$. Using only classic color models for foreground and background is sensitive to any similarity of appearance between these two parts. Adding the multi-view color term helps to take into account the other viewpoints and we see that some regions

Figure 4.6: **Segmentation results**. Results from the multi-view segmentation for four scenes. As the segmentation might be multi-labeled, we show the results as depth maps.

that were erroneously part of the foreground model, are now less likely to be coherent in multi-view. We also see that multi-view links allow the propagation of good segmentation results across viewpoints. More segmentation results are shown in Figure 4.6.

4.5 Pre- and post-processing

Achieving high quality segmentation requires a few pre-processing steps, which we describe below. We also explain the matting and background image creation steps, required for rendering. Recall that the input to these steps is the multi-view stereo reconstruction with calibrated cameras, and the user-defined thin structure geometry.

4.5.1 Pre-processing

Our multi-view segmentation approach assumes that the thin structures are not reconstructed. In some cases, modern reconstruction algorithms create spurious geometry in the thin structure supporting surface $\mathcal{S}$, typically filling the space between them or even reconstructing small parts of the structure. To allow high-quality segmentation, we need to remove this reconstruction, bringing our input data into the canonical form expected by the segmentation.

Removing reconstruction in the thin structure geometry. To identify the vertices of the proxy that comes from superfluous reconstruction (e.g., false surfaces between

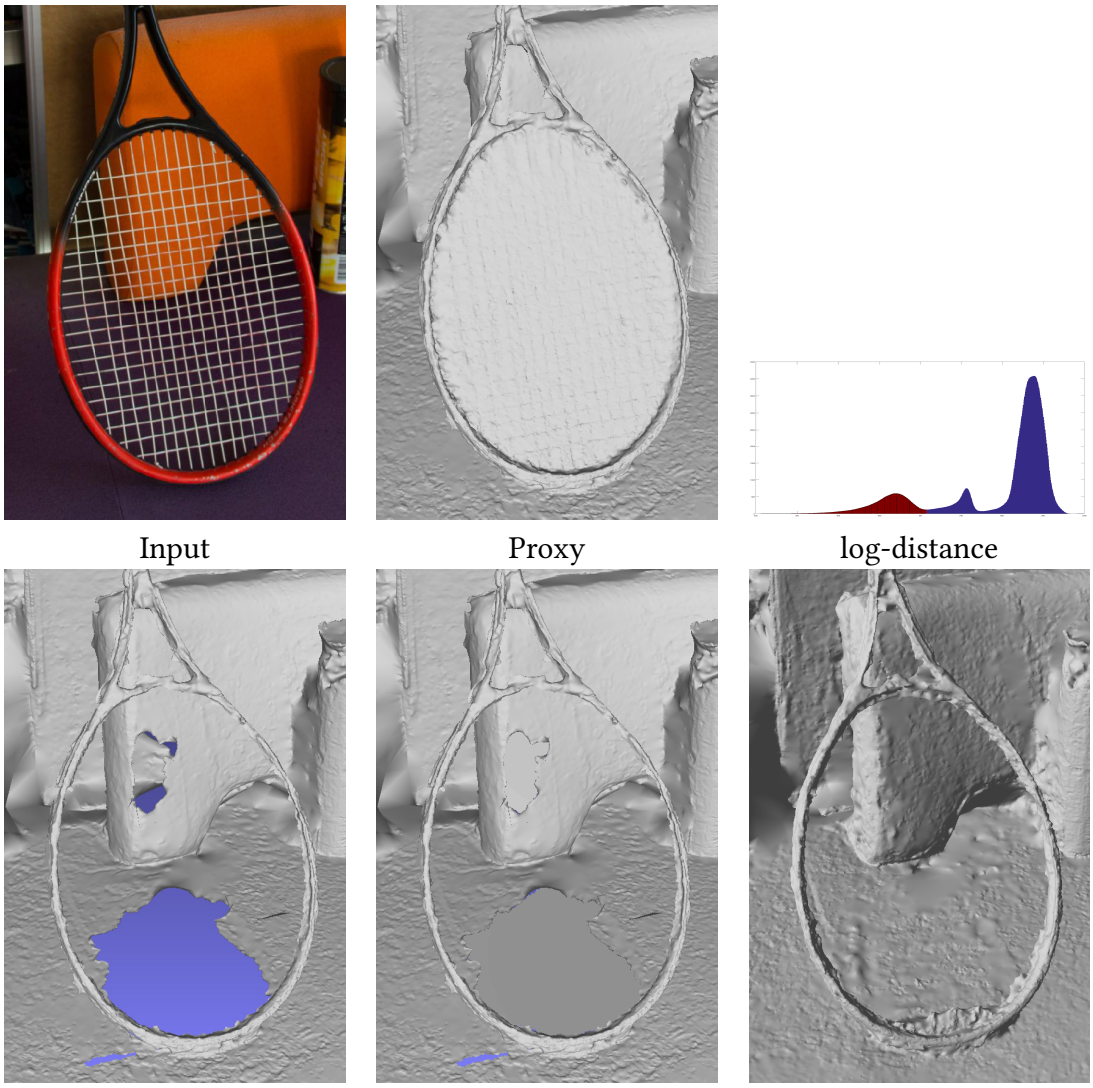


Figure 4.7: **Depth cleaning**. Result of the depth pre-processing for regions with spurious reconstructed geometry in the geometry of thin structures. The histogram shows the log distances distribution between the supporting geometry and the proxy vertices. The closest vertices (red bins) are removed during the cleaning step while the rest of the proxy (blue bins) remains untouched. The filling step provides a planar approximation, which is enough for the segmentation. The final geometry used for rendering, shown at the bottom right, is obtained after the segmentation using thin-structure free images as described in section 4.5.2.

thin structured), we first compute a smoothed histogram of the log point-to-mesh L2 distance between the proxy vertices and the thin structure geometry. Then we remove all the vertices for which the log-distance is smaller than the value associated to the

first valley (see Figure 4.7). Such removal can reveal holes in the proxy that need to be filled to have a depth available behind the thin structure geometry. We fill those holes by fitting local planes.

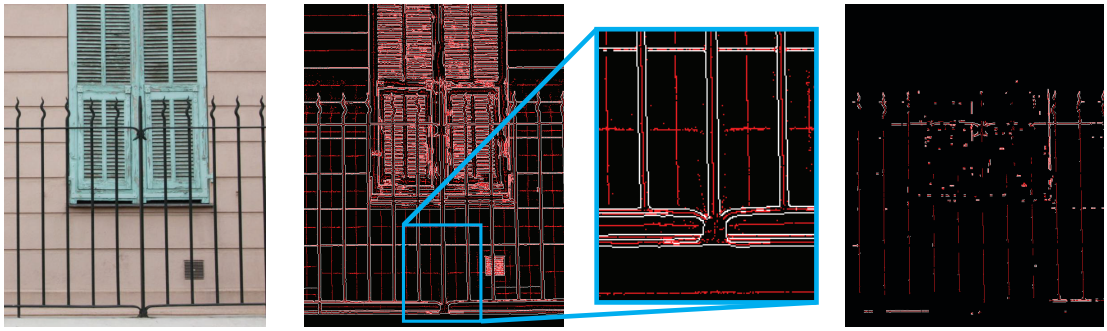


Figure 4.8: **Thin structure candidates for a given input image**. Thin structures are marked in red, between detected edges in white. We filter these thin structure candidates using multi-view information. To initialize the multi-view segmentation, we select thin structure candidates p that have $P_k^{\text{mv}}(p) > 0.9$.

Detection of thin structure candidates. We obtain first candidate points for thin structures by detecting edges in the images [17]. Candidate points are generated between the edges. This is illustrated in Figure 4.8: the edges detected are in white, and the generated candidates points are in red. Since a large number of such candidates can be generated, we filter the result using multi-view information. More specifically, we initialize a thin structure candidate to be at layer k if $P_k^{\text{mv}}(p) > 0.9$ for any $k < N_p$. Every other pixel is initialized with the back-most layer.

4.5.2 Post-processing

In post-process, we need to extract the foreground and an alpha matte of the thin structures, create the background image corresponding to each input view, and create the clean background geometry. Before matting, we remove small spurious outliers by removing connected components with size less than 0.2% of the image size.

Alpha matting. The tri-map for the alpha matting step is obtained by first dilating the foreground label maps to create the uncertain region and eroding to determine the

Figure 4.9: **Post-processing**. From the thin structures segmentation (bottom left) of the input images (top left), we perform single image matting (middle bottom). We also use the segmentation as visibility information to reproject content from other views to inpaint the thin structures (middle top). These thin-structure free images are used to reconstruct the scene a second time (bottom right), providing better quality than the initial reconstruction in occluded regions (top right).

foreground. The accuracy of the tri-maps is important to the success of the subsequent steps, but also depends on the matting algorithm used. We use the approach of He et al. [49] for matting, which provides adequate alpha mattes for rendering. An example of alpha matte result is shown in Figure 4.9.

Background generation. We use the median images, updated with the visibility information from the segmentation, as background in the regions labelled as thin structures. We also run a Poisson editing step to correct for small differences in color levels. A small number of pixels are black in these images, corresponding to regions occluded by the thin structures in all images; we apply standard single-image inpainting to fill these holes using a variant of PatchMatch with the "occurence" term of Kaspar et al. [60]. Once we have generated these color-balanced background images, we perform a fresh 3D reconstruction step with the thin structures removed. This step improves the quality of the background mesh (Figure 4.9).

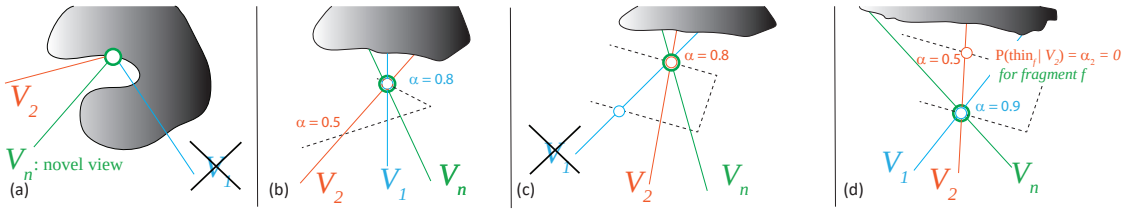


Figure 4.10: **Rendering algorithm**. (a) Traditional ULR rendering: For fragment f (green circle) we ignore view V_1 because there is an occlusion. (b) In our case, we have multiple values of α from different views, that blend with the background. These are interpreted as probabilities of being a thin structure at this depth, thus in this example $P(\text{thin}_f|V_1) = \alpha_1 = 0.8$ and $P(\text{thin}_f|V_2) = \alpha_2 = 0.5$. (c) If for an input view the depth map is in front of the current fragment, the view (V_1 here) is ignored as the view gives no information about the fragment. (d) If for an input view, the depth map is behind the current fragment, the view (V_2 here) estimates there is no thin structure at the fragment location with $P(\text{thin}_f|V_2) = 0$.

4.6 Rendering

Our rendering algorithm is based on the Unstructured Lumigraph (ULR) [14] with soft visibility [33], operating on a per-pixel basis. Standard ULR precomputes a depth buffer for each input view. During rendering, a first pass renders the depth to the frame buffer, and for each fragment, we perform a depth test for each input view with the precomputed depth. If the test fails the fragment is discarded Figure 4.10(a). If it succeeds, the ULR weights w_i are computed [14]. These express the match in angle and distance between the novel and input views, and used to blend the corresponding colors from the input images.

Our scenes are more complex. The depth map for each input view is computed using the background geometry, and then augmented with the information provided by the segmentation. Each pixel on a thin structure has a depth corresponding to the supporting geometry layer determined by the segmentation, and an alpha value computed by the matting process (Figure 4.10(b)). There are two main issues: first, the supporting geometry of the thin structure (dashed lines in Figure 4.10(b)-(d)) is semi-transparent, and thus requires a specific rendering algorithm. Second, we need to define how to combine the view-dependent alpha values in the novel view.

For the first issue, we adapt the depth peeling algorithm [34], run in back-to-front order. This consists in progressively rendering each layer of depth with a “less-than” depth test and alpha blending with the previous layer. First, we render the clean background proxy

using standard ULR, then perform depth peeling using the supporting geometries.

To define how to combine view-dependent alpha, we propose an interpretation of α values as conditional probabilities of having a thin structure along the viewing ray of a pixel in each input view. $P(\text{thin}_f|V_i)$ is the conditional probability that a fragment f is on a thin structure given view V_i. We thus assume $\alpha_i = P(\text{thin}_f|V_i)$; we interpret ULR weights [14] as a prior probability of each view V_i:

$$P(V_i) = \frac{w_i}{\sum_k w_k}. \tag{4.10}$$

We will apply Bayes rule to compute the overall probability of a given fragment being on a thin structure, at a given layer of the depth peeling algorithm, which can be seen as a multi-view filtering of the segmentation results. Then we use this probability as the α_f value for blending the weighted ULR colors from the input views for fragment f (see algorithm 2 for the details):

$$\alpha_f = P(\text{thin}_f) = \sum_i P(\text{thin}_f|V_i)P(V_i) = \frac{\sum_i \alpha_i w_i}{\sum_i w_i} \tag{4.11}$$

There are two cases which need to be treated as specific depth tests. First, if a fragment of a thin structure from another input view V_j is in front of the current thin structure fragment f, then V_j is ignored by setting $w_j = 0$ (e.g., Figure 4.10(c), V_1 is ignored), since it cannot provide information about a fragment behind it. The second case is if there is a thin structure at a depth behind the fragment f in an input view. Since the segmentation has placed it behind f, the conditional probability that the thin structure is at the depth of f is 0; we thus set $P(\text{thin}_f|V_i) = \alpha_i = 0$. However, we include the input view in the blend of colors to be conservative. This approach is related to the consensus voting approach in Soft3D [95]. Since we have surface-based geometry, instead of performing volumetric accumulation, we can directly use the probabilities computed to make a decision for each fragment, which avoids visual artifacts far from the input cameras.

4.7 Results and comparisons

We provide comparisons with related work and then discuss limitations. When judging comparisons, one should consider that our results benefit from the user-defined thin

Figure 4.11: Rendering results for 9 different scenes from novel views far from the input cameras. From left to right: rendering results for Selective IBR [90], our improved version of ULR [14], a re-implementation of Soft3D [95], the inpainted background and the top view showing the novel camera, and our result.

Algorithm 2: Multi-depth thin structure rendering algorithm.

Input: Images with depth, background, and thin geometry
Result: Novel View
Render clean background with standard ULR into C_{current};
for *each depth peel layer, back to front* **do**
    Render alpha with depth tests in α_f;
    Render color with ULR weights in C_f;
    Blend with previous layers : $C_{\text{current}} \leftarrow C_f \cdot \alpha_f + (1 - \alpha_f) \cdot C_{\text{current}}$;

geometry, but this information only could not be directly used by these previous algorithms.

4.7.1 Results

We ran our approach on an Xeon E5-2650 PC. For each scene we took between 10 and 30 photos. The outdoors scenes were taken with a DSLR camera, and the indoors with an iPhone 6S. We use the RealityCapture software [101] for camera calibration and 3D reconstruction. Note that we assume that a background depth is reconstructed for all pixels in an image. To treat a scene with a sky background for example, a bounding box could be fit to the scene to provide "far depth" for sky pixels. The user specifies the supporting geometry by providing correspondences points in a multi-view interface to fit a basic 3D primitive and then manipulates the primitive control points in a 3D window. Our current interface handles planar segments and cylinders. Please see the video for an example interaction session[1].

The offline pipeline (excluding reconstruction steps) took from 5 minutes for the smaller scenes ($\sim$ 10 images) to 20 minutes for the larger scenes ($\sim$30 images). The rendering runs at 60 fps on a 1080p display with all datasets using a GTX 1080.

Our approach allows us to capture complex shapes which are not necessarily purely repetitive, for example in the *Stairs* scene. Small artifacts in the segmentation are often alleviated by the blending step of IBR. We show the results of our inpainting step and rendering from novel viewpoints in Figure 4.11. The *Stairs* and *Rolland* scenes contain multiple layers of thin structure depth, and the Balcony scene is a cylinder with multiple depths. The supporting geometry could in theory have any form, as long as it can be

[1] http://www-sop.inria.fr/reves/Basilic/2018/TDDD18/

reasonably represented by a mesh.

4.7.2 Comparisons

We present comparisons on rendering and segmentation. For rendering, please see the accompanying video, the improvement over previous methods is much more evident during interactive free-viewpoint navigation.

Rendering. Our approach allows free-viewpoint navigation to regions quite far from the original cameras. We thus focus our comparisons on other methods that allow interactive free-viewpoint navigation, in particular Unstructured Lumigraph Rendering (ULR) [14] and Bayesian IBR [90] (which subsumes the work of Chaurasia et al. [21]). For these comparisons we use our re-implementation of ULR, which in contrast to the original version, uses *per-pixel* weights based on the geometric proxy from MVS reconstruction, and the original implementation of Bayesian IBR [90]. We show results for several scenes and novel viewpoints, far from the input cameras. Our method greatly improves over previous methods, since those methods incorrectly project thin structures onto the background geometry.

In addition to these free-viewpoint methods, we compare to the recent Soft3D IBR method [95]. We use an re-implementation of the complete Soft3D method provided by P. Hedman of UCL[2]. As can be seen in the video, Soft3D produces high quality results for certain capture configurations (e.g., all cameras in a plane) and when interpolating camera positions. When moving away from the input cameras, blurring artifacts appear due to the depth discretization. These are even more visible for thin structures. Note also that rendering a single image with Soft3D takes seconds, compared to the real-time behavior of our method.

Segmentation. To our knowledge, no previous method has addressed thin structures for image-based rendering. As a result, we compare with the most closely related methods, which are not tuned for our data. For completeness, we also compare to previous lattice-detection methods in Figure 4.14. The results show that such methods either fail to find parts or even most of the thin structures, or sometimes only partially succeed

[2]We provide a comparison of our results for the Soft3D algorithm and the original implementation on the Museum scene [21] in an additional video. The general image quality is similar, even though the original implementation is slightly sharper.

Cathedral House

Figure 4.12: **Results of video de-fencing method**. This is the result obtained on *House* and *Basilic* datasets with a method based on motion estimation by Yi et al. [136]. reliable matches between consecutive images are unlikely to be computed in a wide baseline setup.

Background Foreground

Figure 4.13: **Comparison with obstruction free photography** by Xue et al. [134]. Results on Cathedral dataset: left image is the background layer and right image is the foreground layer. Optical flow based methods are unlikely to handle wide baseline configurations.

for some, but not for all images in the multi-view dataset. This is unsuprising given our richer input (camera calibration, 3D reconstruction and user-defined thin geometry); The comparison is provided to show that these methods cannot solve our problem.

We also compare to automatic fence segmentation [136] in Figure 4.12. The authors kindly ran their code on our dataset, however since our data are not continuous vide-

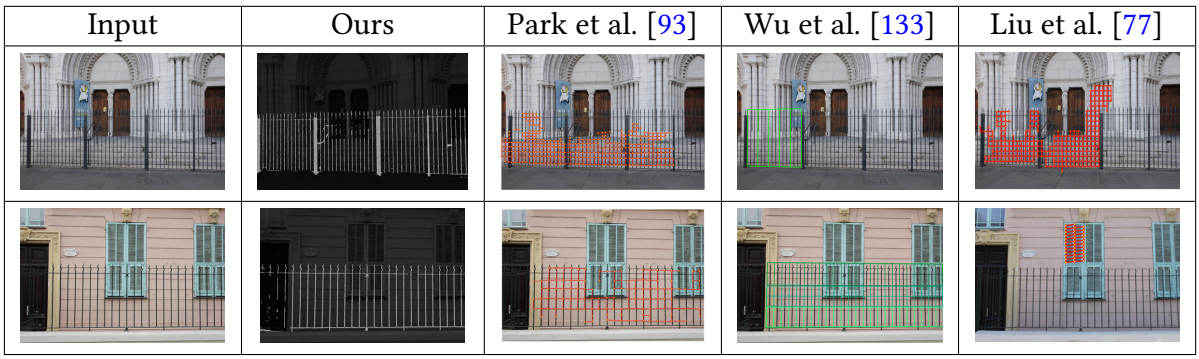


Figure 4.14: **Lattice detection methods**. We show comparisons with our segmentation of thin structures for two images of the dataset.

oframes their method cannot perform spatio-temporal refinements based on frame-by-frame optical flow. Thus the results shown are the "initial segmentation" described in the paper. If the input requirements for this method are respected, the results would be better. Similarly, we compare to the method of Xue et al. [134] run on a crop of the input images in Figure 4.13. Again this method expects a smaller baseline between images, and thus cannot be used for our target scenes.

Comparisons to Lattice Methods Some image *de-fencing* methods (e.g. [94]) rely on the redundancy of thin structures to detect fences in images. This type of approaches is unlikely to succeed with see-through objects we consider in our scenes. We provide in Figure 4.14 comparisons with lattice-detection methods which have variable levels of success. These methods either fail to find parts or even most of the thin structures, or sometimes partially succeed for some but not all of the images. All results shown were either provided by the authors of the original articles or created using code available online.

These negative results are unsurprising since our input is much richer, as we have camera calibration and 3D data linking the multiple input images as well as the user-defined thin geometry. We provide the comparison simply to show that these methods are not able solve our problem. Other alternatives such as applying these algorithms on the median images of the thin structure geometry are unlikely to improve results since the blurred and composite results in the background regions make feature detection much harder.

Figure 4.15: **Limitations**. (a) At extreme grazing angles, the infinitely thin assumption is insufficient. (b) Very small structures in input images can be missed by our approach (see rendering in (c)).

4.7.3 Limitations

Our method fails when the background has the same color as the foreground in all views. This is simply due to fact that there is no view available to leverage the color ambiguity between the different layers. Also, our simple thin geometries are implicitly assumed to be infinitely thin; this works well for most cases, but becomes problematic at extreme grazing angles such as in Figure 4.15(a). The ability to detect thin structures also depends on the resolution of the images with respect to the thickness of the structures. When the structures are too thin, our approach can fail, as seen in Figure 4.15(b)-(c).

In all our examples, the supporting surfaces produced were accurate enough to produce good quality results. Consequently, we did not perform in-depth robustness analysis. Very imprecise user input could result in inaccuracies for the supporting geometry, affecting the segmentation quality because of the larger number of incorrect multi-view links. For such input, the surface estimation could be refined between segmentation steps, e.g., using intermediate segmentation results to find dense correspondences.

4.8 Conclusions and future work

We have presented a new method to treat the hard problem of thin structures for image-based rendering. Our approach is a multi-layered segmentation algorithm which ex-

ploits multi-view information and 3D reconstruction to provide segmentation of thin structures. Our end-to-end solution allows us to produce results showing significant improvement over previous methods in terms of rendering quality, allowing common scenes with thin structures to be used in a free-viewpoint IBR context for the first time.

The key aspect of our work is to leverage multi-view information to overcome the ambiguities in the hard case of thin structure segmentation. Using other priors as discussed above will allow this approach to generalize to other structures in terms of complex geometry and varying frequencies of the repetitive patterns.

However, they are many opportunities for future work, for example concerning the robustness and the scalability of the method. The major limitation for treating bigger scenes is the user interaction. Automating the detection of supporting geometries could be achieved with learning-based solutions.

Recent high-quality segmentation deep learning networks (e.g., [73]) could potentially be adapted and trained to identify thin structures, although generating the training data is a challenge. Similarly, deep learning has been used to find parameters of simple geometric structures [88]. With suitable knowledge of semantics and the correspondence between supporting primitive type and thin structure, such an approach could be developed to allow primitive fitting based on images, potentially combined with an optimization step [91]. Another interesting challenge to solve is the interaction of thin structures and vegetation which makes the problem much harder.

In conclusion, in this chapter we have seen how exploiting guided multi-view information allows to widen the type of scenes IBR is able to capture and render. We will see in the next chapter how extending the casual capture to videos could broaden the scope of IBR even further by dealing with time-dependent scenes.

Chapter 5

Practical video-based rendering of dynamic stationary environments

Contents

5.1 Introduction

A major advantage of Image-Based Rendering (IBR) algorithms is that they provide high-quality free-viewpoint navigation of a scene captured in *casual* manner, i.e., by simply moving a single camera or smartphone around the scene. We have seen in the previous chapters that the only requirement is to take enough pictures for structure from motion (SfM) [117] to calibrate the cameras, and multi-view stereo (MVS) [43] to approximately reconstruct scene geometry. Recent IBR methods [95, 52] provide free-viewpoint navigation of unprecedented quality. However, they suffer from a major limitation: all content in the scene is static, and small motion such as waves rippling on a beach or flames in a fireplace cannot be captured, nor reproduced for free-viewpoint navigation.

We introduce a video-based rendering method which allows such free-viewpoint navigation, handling *stochastic* dynamic phenomena such as waterfalls, streams, small waves or fire. Our method preserves the advantage of casual capture since it typically requires a single smartphone and a cheap tripod, a handful of videos of the scene, and the same number of photos as for SfM/MVS. This is in contrast to past video-based rendering solutions which involved complex *synchronized multi-camera* systems and typically focused on human motion [19, 25, 127] or only allow limited *transitions* between views [8, 124].

The main challenge for our casual capture setup comes from the fact that we blend *unsynchronized* videos from different viewpoints, thus the different views of the dynamic phenomenon are not photo-consistent. This raises several challenges. First, naive blending between input videos in the novel view will cause excessive blurring, and generates unsightly temporal discontinuities, due to the multi-view nature of our data and the fact that we loop short input sequences over time. We combine an extension of the Laplacian blending in time and standard blending-field-based IBR to overcome these issues in our multi-view spatio-temporal context. Second, the phenomena we treat can be volumetric and semi-transparent (fire, waterfalls etc.); the unsynchronized nature of the videos precludes the use of SfM/MVS, and tomography-style reconstruction [137] would fail since opacity and the resulting volume cannot be reliably estimated. Instead, we present a solution that first approximately localizes the effect in 3D space, provides per-view geometry for reprojection, and enables alpha matte estimation.

We combine the spatio-temporal blending, localized dynamic geometry and matting into an interactive free-viewpoint rendering algorithm including dynamic phenomena

such as waves, waterfalls or fire. Our results show an effective solution to allow free-viewpoint navigation in an image/video-based rendering context, greatly enhancing realism and the sense of immersion in captured scenes.

Our blending strategy, per-view geometry and alpha mattes allow free-viewpoint rendering with stochastic dynamic effects captured with unsynchronized videos. We show results on phenomena such as fire, waterfalls or rippling waves in a seaside scene, bringing these scenes to life. Our method allows interactive free-viewpoint navigation while maintaining the high-frequency visual components of these effects.

5.2 Related work

Image-based rendering algorithms [14, 21, 52, 95] only require a sparse set of unstructured photos from a scene to provide high quality, free-viewpoint rendering, often in real-time. Each such method makes some trade-off between exploration capabilities, ease of capture and complexity of treated scenes. Similarly, our method strives to allow free-viewpoint navigation with dynamic content, thus "bringing the captured scene to life"; inevitably we also are confronted with different trade-offs.

Most image-based rendering methods (e.g., [14, 21, 53]) rely on a geometric representation of the scene that is used to reproject the input photos onto the novel view, and assume photo consistency between input images. Recent IBR methods have attempted to overcome inaccuracies in reconstructed geometry in various ways, e.g., using per-view representations [21, 53], learned blending weights [52], or modeling uncertainty via a volumetric representation [95]. In our context, the photoconsistency assumption is broken since we work with unsynchronized videos which witness events corresponding to different moments in time with possibly different appearance. However, our technical choices are often inspired by those developed for IBR, e.g., per-view geometry or volumetric representations used for the dynamic effects.

Several methods tackled the problem of human capture [19, 25, 127, 118]. However, to retrieve accurate capture of human motion, they have to rely on multiple systems with synchronized videos. Moreover, by restricting the use case, they can take advantage of human body-based priors such as tracking of specific human parts or global template fitting. In our context, we aim to keep the same level on capture complexity as traditional IBR hence our choice for using unsynchronized videos.

5.2.1 Video looping and compositing

Videos are a natural way to capture the dynamic nature of a scene. However, a video only provides a fixed segment in both time and space. Therefore, many methods have been proposed to extend these limitations by creating endless video streams and combining the information from multiples videos.

Video Textures [104] create an infinitely-long non-repeating video stream from one static video clip. By finding probabilities of transitions between frames, the method produces a random video player for a wide variety of motions. Panoramic Video Textures[4] use a rotating camera to capture dynamic effects with a wider view angle than a single video. This method relies on graph cut compositing to generate dynamic effects for the whole panorama, but the viewpoint position remains fixed at runtime. The work of Bai et al. [7] allows users to selectively remove large scale motion in a video. This allows the creation of *cinemagraphs*, which are photos with some subset of pixels having a looping motion. The method of Liao et al. [72] automatically captures different levels of dynamism in a given static video. This enables the generation of a whole spectrum of video loops based on the variety of looping subregion periods. More recently, He et al. [50] demonstrated the creation of a single gigapixel panorama video loop from a structured but unsynchronized capture of videos, providing very detailed rendering and handling an even wider range types of motions.

All these purely video-space looping methods allow the capture of virtually any kind of looping motion, without severe limitations on the content. However they have either a fixed field of view, or rely on complex graphcut optimizations that cannot run in realtime. As we shall see later (Sec. 5.4.2.2), we can create loops of our stochastic content using a simpler and cheaper solution, for use in an interactive free-viewpoint renderer.

5.2.2 Exploring video datasets

Ballan et al. [8] use a sparse, unstructured and contemporaneous video capture setup, which they can synchronize. By adjusting the transition points at runtime, and by modeling the static background and the dynamic foreground separately, the method enables navigation around a performer. In a similar manner, the Videoscapes method [124] relies on a sparse but huge amount of unstructured and unsynchronized videos to explore a scene, by identifying transitions between moving video clips. Both methods try to re-

duce the impact of the transition between clips on the user experience. In contrast, we focus on free-viewpoint navigation rather than transitions, and achieve this by limiting the scope of scenes we treat.

Interactive Viewpoint Textures [69] use a semi-structured and unsynchronized video capture setup, allowing free viewpoint navigation along a path. This work differs from ours since it relies on optical flow rather than a geometric representation of the scene, but is much more limited for navigation. In addition, optical flow cannot be successfully computed on our wide-baseline input.

5.2.3 Multi-video dense reconstruction

Several methods overcome the challenges of dynamic scenes by using dense or constrained capture setups. The method of Gregson et al. [46] uses a dense capture setup of synchronized videos to reconstruct turbulent fluids. It uses light-computed tomography and can render the captured fluids without explicit 3D volumes. Zang et al. [137] use only one sensor but capture thousands of X-ray projections. They show very high quality reconstructions of solid objects degrading over time. Instead of relying on reconstruction, Okabe et al. [89] use a sparse setup of synchronized videos to synthesize dynamic effects, in terms of both appearance and volume. They alternate between appearance transfer based on histogram matching on steerable pyramids coefficients, and volume optimization in an expectation maximization fashion.

5.3 Overview

Before addressing the actual compositing algorithm, we briefly discuss our capture procedure, using a set of videos with fixed positions (10-20 per scene) and photos ($\sim$20-80 per scene) used to estimate 3D information. A median image is first extracted from each input video. These images are combined with the input photos to reconstruct the scene geometry, using an off-the-shelf structure from motion and multi view stereo algorithm [101]. However, the scene might require some manual intervention such as fitting a global water plane[1]. After this step, the input of our method is a set of calibrated cameras corresponding to both the input videos and the input photos, as well as a reasonable approximate geometry for the static scene parts.

[1]Scenes *Seaside*, *Cave* and *Beach* required manual water plane fitting.

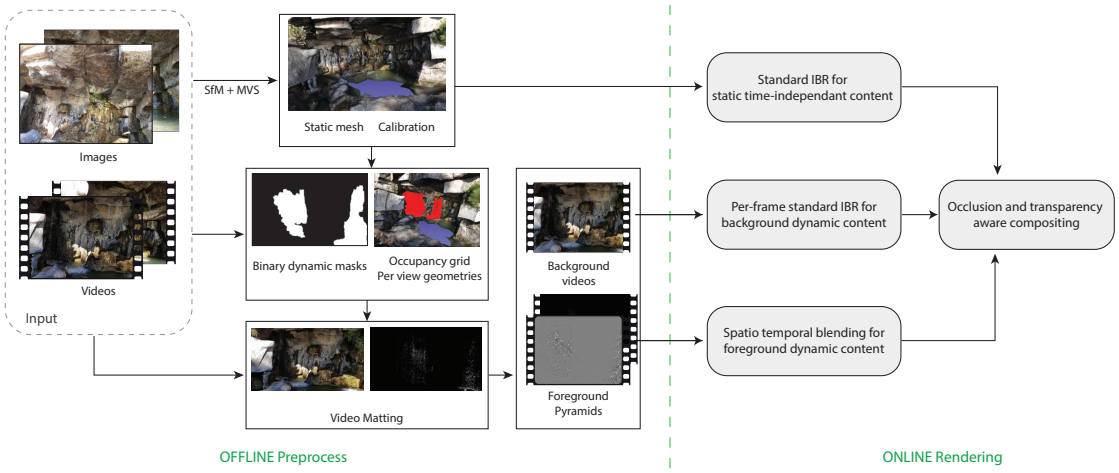


Figure 5.1: **Method overview.** Using a multi-view dataset composed of unstructured images and unsynchronized videos, we first perform standard SfM and dense reconstruction. However, to render the dynamic effects without spatio-temporal seams, we first have to extract, in a preprocessing step, a scene representation which localizes the relevant dynamic content both in image space and in 3D space.

We start with the simplest case of an opaque surface such as a water basin with rippling waves, where we assume known geometry and location of the dynamic effect. There are two kinds of spatio-temporal discontinuities in novel views, those due to the non-looping nature of the input videos, and those due to the lack of synchronization between cameras. We introduce a spatio-temporal blending approach, by extending Laplacian blending to time, allowing temporal-frequency-based IBR, and propose a simple yet effective solution for video looping.

For more complex cases such as a waterfall or flames of a fire, the dynamic effect is localized in 3D space in a potentially time-varying manner. To allow reprojection into a novel view, we must estimate a 3D supporting geometry, and since these cases often involve semi-transparent elements, we need to develop an effective matting strategy. We introduce a solution to motion localization using frequency-based analysis, and compute a voxelized visual hull of the dynamic effects. The voxels are used to extract per-view geometric proxies, allowing reprojection of the input videos into the novel view. An overview of our solution is shown in Figure 5.1.

5.4 Seamless spatio-temporal blending

Our capture of stochastic phenomena introduces two main kinds of discontinuities in the space-time domain: those implied by the creation of arbitrarily long videos from short inputs and those due to space and time varying compositing of the unsynchronized multi-view videos. We seek to make these boundaries seamless to avoid unpleasant visual discontinuities in the final free-viewpoint navigation output. To do this, we extend and adapt Laplacian blending techniques to the spatio-temporal context of *unsychronized* multi-view videos, and specifically free-viewpoint Video-Based Rendering (VBR).

In this section, we introduce the background and the basic principles of our temporal Laplacian pyramid. For now, we assume the simplest case, where the 3D geometry of the scene is known and all its elements are opaque. This could be for instance the opaque basin filled with muddy water, with the water surface approximated by a plane. In section 5.5, we lift these restrictions and discuss how to localize dynamic phenomena is space, allowing us to treat cases such as fire and waterfalls that occupy a volume in space and are semi-transparent.

5.4.1 Origin of the boundaries

To allow users to explore the scene without a time limit, we generate arbitrarily long video content from our input sequences that last between 10 and 15s. The solution is to make each sequence loop, which introduces a temporal boundary after each repetition. Then, because no single input sequence sees the whole scene, we create the output by stitching several input videos together, thereby creating spatial boundaries. Since users freely control the output viewpoint, these boundaries change over time. From this perspective, the output video generated by our approach is a patchwork made of the input sequences with complex space-time boundaries separating them. As shown in the companion video, naively stitching the input sequences generate unpleasant discontinuities. Next, we explain how we produce a seamless composite.

5.4.2 Seamless compositing

We propose compositing algorithms inspired by the multi-scale blending approach of Burt and Adelson [15]. We first briefly review this technique before describing how we adapt it to our context.

Background on multi-scale blending Burt and Adelson described how to use image pyramids to blend two images seamlessly [15]. Their algorithm is based on the Gaussian pyramid $\mathcal{G}$ and the Laplacian pyramid $\mathcal{L}$ which are defined as follows for an image I and n_{L} levels.

$$\mathcal{G}[I]_0 = I \tag{5.1a}$$

$$\text{for } 0 < \ell < n_{\mathrm{L}}, \quad \mathcal{G}[I]_\ell = \mathrm{reduce}(G \otimes \mathcal{G}[I]_{\ell-1}) \tag{5.1b}$$

$$\text{for } 0 \leq \ell < n_{\mathrm{L}} - 1, \quad \mathcal{L}[I]_\ell = \mathcal{G}[I]_\ell - \mathrm{expand}(\mathcal{G}[I]_{\ell+1}) \tag{5.1c}$$

where G is a Gaussian kernel, $\otimes$ the convolution operator, and reduce and expand the operators that half and double the image resolution respectively and resample.

Given two images I_1 and I_2, and a binary mask $\mathcal{M}$ which identifies a region of I_1 to be pasted into I_2, the algorithm builds the Laplacian pyramid of the output composite:

$$\mathcal{L}[O]_\ell \quad = \quad \mathcal{L}[I_1]_\ell \times \mathcal{G}[\mathcal{M}]_\ell \quad + \quad \mathcal{L}[I_2]_\ell \times (1 - \mathcal{G}[\mathcal{M}]_\ell) \tag{5.2}$$

where all the operations are performed per pixel. The final output O is reconstructing by collapsing its Laplacian pyramid, that is, repeatedly expanding and summing its levels.

Intuitively, this operation blends each frequency band at "the right scale", i.e., high frequencies are over a very short distance, thereby avoiding potential ghosting artifacts, while low frequencies are slowly cross-faded to prevent introducing unsightly discontinuities.

We introduce an extension to this method using video volumes, considering time as the main dimension instead of space. Each level of the Laplacian video pyramid is itself a video representing one temporal frequency band of the input video. However, instead of keeping levels of different length, we use the equivalent representation of an expanded pyramid where the 1D expand operator [15] has been used to upscale each level in time up to the input length. Such a scheme allows a better visualization of the contribution of each level, since the input video simply becomes the per-frame per-pixel sum of all the levels (see Figure 5.2 and Figure 5.3).

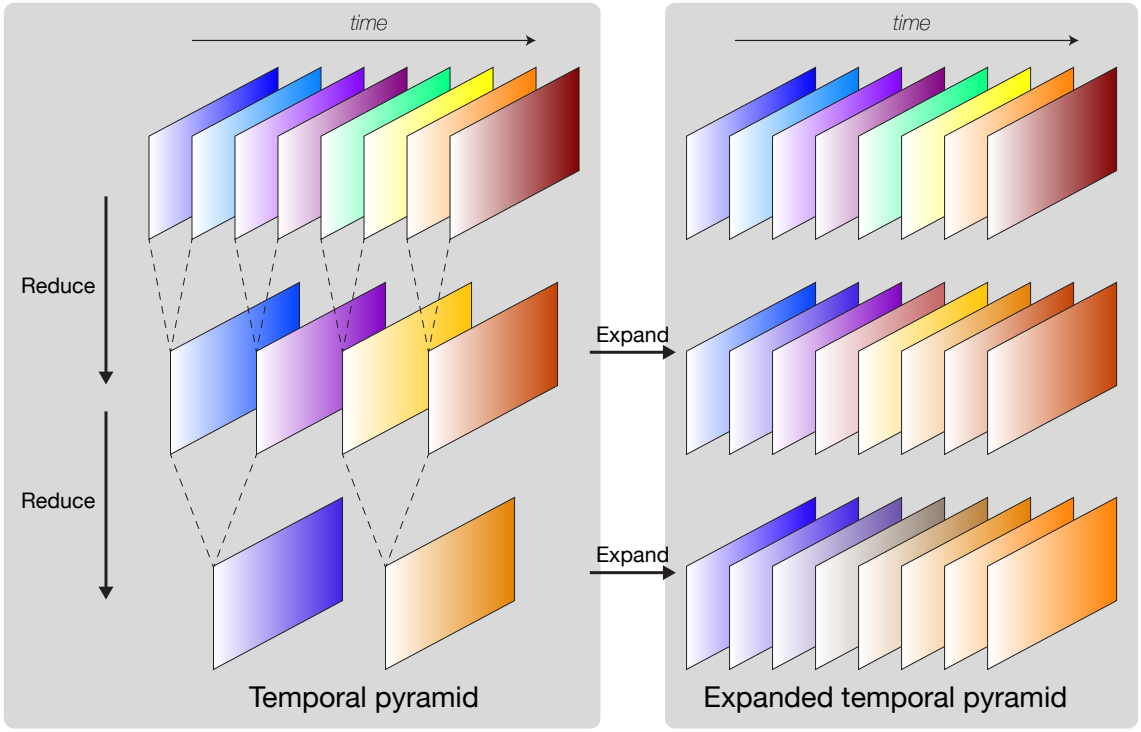


Figure 5.2: **Expanded Temporal Laplacian Pyramid.** Given an input video, a pyramid is computed with each level having half the number of frames than the previous level (left). Each level is then expanded to have the same length as the input (right). Figure 5.3 shows an example of such decomposition.

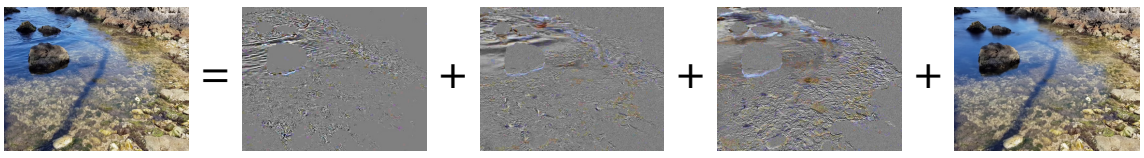


Figure 5.3: **Sample frames from an expanded temporal pyramid.** Figure 5.2 illustrates how we generated these frames.

5.4.2.1 Spatio-temporal blending

A direct extension of IBR to videos would consider each frame independently, where at each frame a new set of input images is available and can be used by any IBR method. However, since the videos are not synchronized, the images are not photo consistent. As a result, a weighted average produces blurred results, losing all the spatial high frequency details of the different inputs. On the other hand, IBR with a sharp blending field, or

similarly view-dependent image stitching, e.g., using the Poisson compositing [96] or image Laplacian blending [15], produces image-space seams when moving the viewpoint because the camera selection creates a "wavefront" on the non photo-consistent content.

Since our videos depict stochastic effects which are stationary in time, photo-consistency heavily depends on the temporal frequency we are considering. Low frequencies correspond to the effect support, which can vary from one viewpoint to another, but overall have the same appearance. In contrast, temporal high frequencies are highly uncorrelated between frames and viewpoints, but they share the same spatial distribution across viewpoints.

To support free-viewpoint navigation, we introduce a spatio-temporal blending strategy, corresponding to multiple IBR blending schemes which are independent per temporal frequency. The key idea is that each different temporal frequency band represents video content which we blend using appropriate IBR blending strategies. In Figure 5.2, we see that each input video V_i is decomposed using a temporal Laplacian Pyramid into n_{L} video-levels $\mathcal{K}[V_i]_\ell$ such that for each frame t:

$$V_i(t) = \sum_{\ell=0}^{n_{\mathrm{L}}-1} \mathcal{K}[V_i]_\ell(t) \tag{5.3}$$

where $\mathcal{K}[V_i]_0$ is a video containing the highest temporal frequency band, while $\mathcal{K}[V_i]_{n_{\mathrm{L}}-1}$ is the lowest temporal frequency residual, i.e., a video where each frame is an approximation of the video average over time (Figure 5.2 and Figure 5.3).

We now describe the blending method for a given frame, thus dropping the t dependency for convenience. For each video i and each level ℓ, the video level textures $\mathcal{K}[V_i]_\ell(t)$ are projected onto the geometry:

$$\hat{\mathcal{K}}[V_i]_\ell = R\left(\mathcal{K}[V_i]_\ell\right) \tag{5.4}$$

where $R(\cdot)$ is the reprojection operator, with respect to the geometry and the novel view, and $\hat{\mathcal{K}}[V_i]_\ell$ is the reprojected content from level ℓ of video i.

We also compute a blending field based on ULR [14] penalties. However we apply a different normalization per temporal frequency. Denoting $w[V_i]$ the weight associated

to a view i for a given fragment, we use a *softmax* normalization scheme:

$$w[V_i]_\ell = \frac{e^{-\lambda_\ell w[V_i]}}{\sum_i e^{-\lambda_\ell w[V_i]}} \tag{5.5}$$

where $w[V_i]_\ell$ is the final blending weight associated to view i and level ℓ. The parameter λ_ℓ controls the blending field sharpness. For the low frequencies, a small λ_ℓ produces a smooth blending field, similar to standard ULR. For the high frequencies, a high λ_ℓ produces a piecewise-constant blending field where smooth transitions have small support on image-space.

These per-frequency blending fields allow us to compute independent ULR results F_ℓ, which can be interpreted as levels of our output temporal Laplacian pyramid. Our final rendered frame F is then obtained by collapsing this pyramid. Since we work with expanded temporal Laplacian pyramids, the collapse is simply the sum of the different levels:

$$F = \sum_{\ell=0}^{n_\mathrm{L}-1} F_\ell = \sum_{\ell=0}^{n_\mathrm{L}-1} \sum_i w[V_i]_\ell \times \hat{\mathcal{K}}[V_i]_\ell \tag{5.6}$$

An example of this spatio-temporal blending scheme is shown in Figure 5.4. Our method is a natural extension of blending-field-based IBR to videos [14]. Indeed, if we consider images as constant videos, all the temporal Laplacian pyramid levels $\mathcal{K}[V_i]_\ell$ for $\ell \neq n_\mathrm{L}-1$ are equal to 0, and the residual level $\mathcal{K}[V_i]_{n_\mathrm{L}-1}$ is equal to the input image.

5.4.2.2 Seamless Video Looping

A naive option to create video loops is to restart the input sequence when it ends (Figure 5.5a). This produces strong visual discontinuities which is not acceptable in our context. A better approach is to introduce some overlap and cross-fade progressively between the end of the sequence and its beginning. While better than naive looping, this raises the question of how fast to cross-fade: too slow and ghosting artifacts appear, too fast and discontinuities come back. Several options based on optimization have been proposed to address this issue, e.g., [7, 72], and they could possibly work in our context. However, they all come at a significant computational cost. Instead, we show that we can make our videos loop while being efficient by using temporal Laplacian blending.

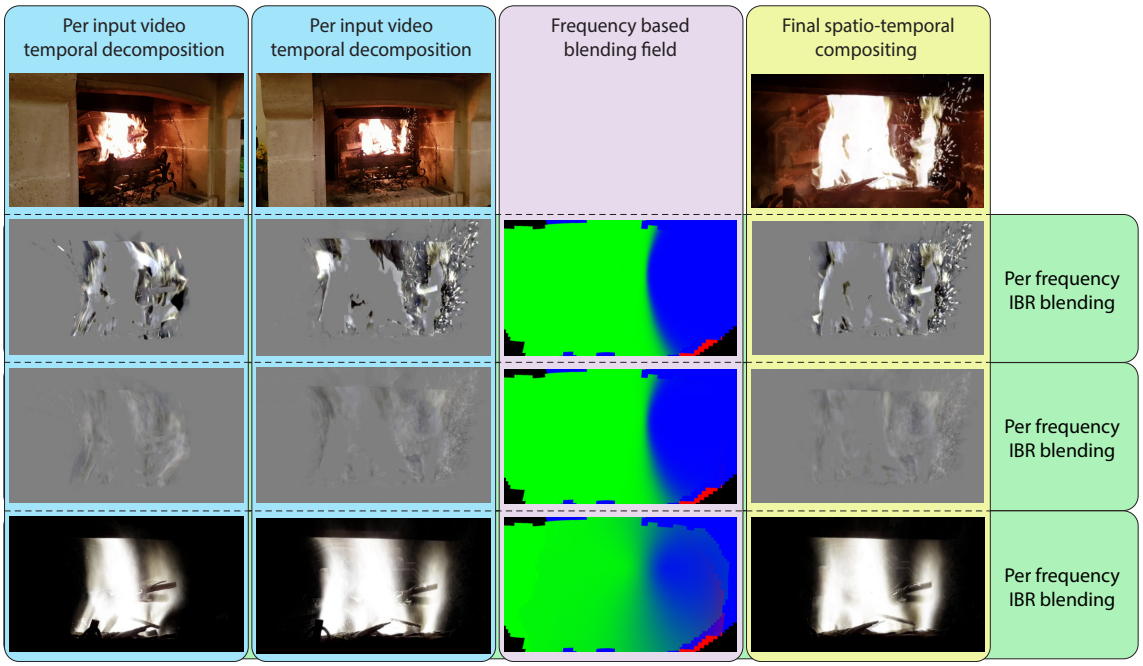
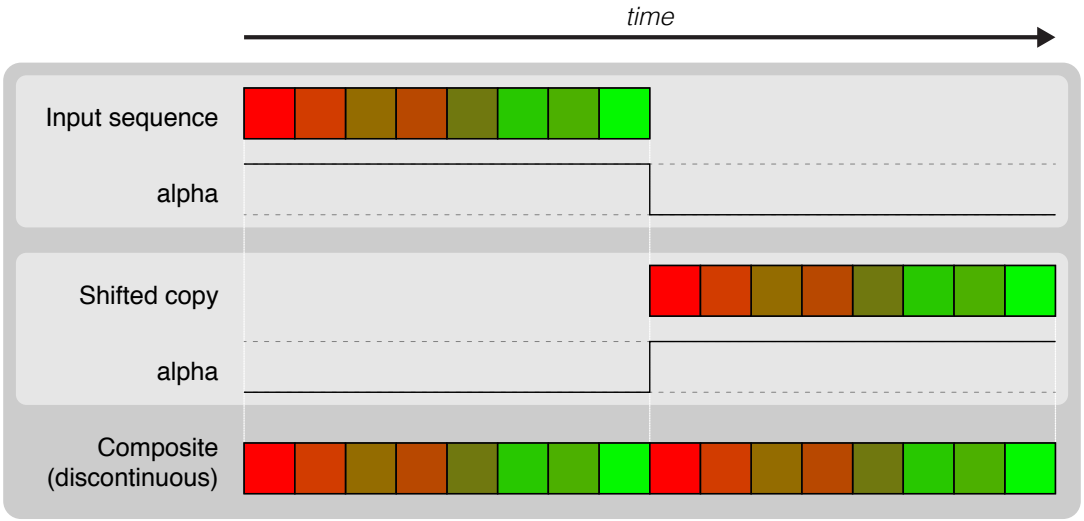


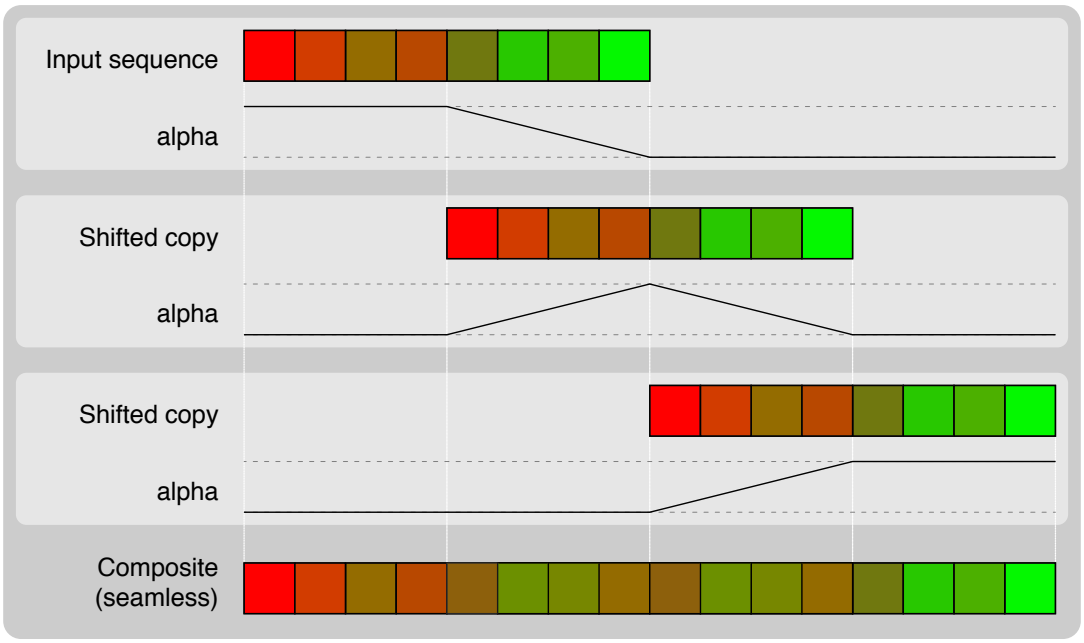
Figure 5.4: **Spatio-temporal blending.** Left two columns: Reprojection of the input video temporal levels. Middle: A different blending field is computed for each temporal frequency, favoring smooth transition for low frequencies. Right: The final frame frame is obtained by collapsing the per-frequency IBR blending.

As illustrated in Figure 5.5b, our approach is to have an overlap corresponding to half the duration of the input sequence. In this setup, there is always a transition happening between the first half and the second half of the sequence, and the output composite is a repetition of this transition. We generate this transition using multi-scale blending in the temporal domain, i.e., we apply Equation 5.1 and Equation 5.2 along the time axis instead of in the image plane, and use a mask that selects the first half of the sequence beginning, i.e., $M = 1$ in the first 25 percent of the sequence. This resolves the question of the speed at which to cross-fade by adapting it to each temporal frequency band. The high frequencies transition sharply in the middle and the lower frequencies cross-fade over longer time spans.

Discussion Our adaptation of multi-scale blending to the time domain implicitly assumes that the observer's perception of temporal transitions is similar to their perception of the spatial transitions for which the original algorithm [15] was designed. Our experiments show that it is true to a first approximation but a slight temporal discontinuity



(a) Naive loop creation (suffers from discontinuities)



(b) Our loop creation (seamless)

Figure 5.5: **Video looping using temporal blending.**

can still be perceived. We hypothesize that this may because the high frequencies at all pixels transition at the same time, which create a compound effect. We found that replacing the sharp mask M by a smooth step over 10 frames produces better results at the cost of minimal ghosting that is imperceptible unless one pauses the video on one of

Figure 5.6: **Video slices.** Natural videos played back to back show a strong temporal discontinuity (middle), while our temporal Laplacian blending scheme produces a seamless video loop (left). Slices are represented with time on the vertical axis and space on the horizontal axis. Displayed slices correspond to the zoomed in segment (left).

these frames.

Implementation Since we blend the beginning of the sequence with its end, we only need to compute one pyramid decomposition. Because of the reduce steps, the levels are aliased and isolating the first half from the second half may not be straightforward. We address this issue by upsampling all the levels back to the input resolution, which removes the aliasing and allows us to split the levels into two halves.

Once looped, our videos are typically $128 = 2^7$ frames long, which corresponds to $n_\mathrm{L} = 8$ temporal frequencies. We use per-fragment ULR based weights [14] with the 4 closest video viewpoints, using an additional term to decrease weights from fragment close to the per-view geometry boundaries to avoid visible occlusions between them. In all our examples we used $\lambda_\ell = 10 \cdot (n_\mathrm{L} - \ell)$.

5.5 Spatio-temporal localization of dynamic elements

In the previous section, we assumed that the entire scene was dynamic and its 3D geometry was known. We now discuss the more general case where the dynamic content is localized in space and its geometry is usually not well reconstructed by multi-view stereo. Typical examples of this are flames of a fire or a water fountain, shown in Figure 5.7. In addition, such dynamic phenomena are often semi-transparent, posing an additional challenge since rendering these effects will require matting.

Figure 5.7: Frames from two videos of a fire (corners), which is a dynamic stochastic phenomena that is localized in 3D. When captured with unsynchronized video and photos, the geometry of the phenomena is not reconstructed (center).

Recall that we have *unsynchronized* videos of a stochastic and typically semi-transparent phenomenon localized in space. Previous methods which capture such phenomena have used multi-camera *synchronized* video setups [46]. In their context, the typical method first estimates volume density (akin to opacity), implicitly estimating the spatial extent of the volume with a tomography-like approach. Such an approach is impossible in our case, since the unsynchronized nature of our videos does not allow estimation of opacity. Instead, we need to invert the order of operations, and first approximate an overall bounding volume of the dynamic phenomenon over time, then provide supporting 3D geometry to allow reprojection of videos and finally estimate per-view per-frame opacity in the form of alpha mattes. This approach allows us to represent semi-transparent regions for phenomena like flames and water streams.

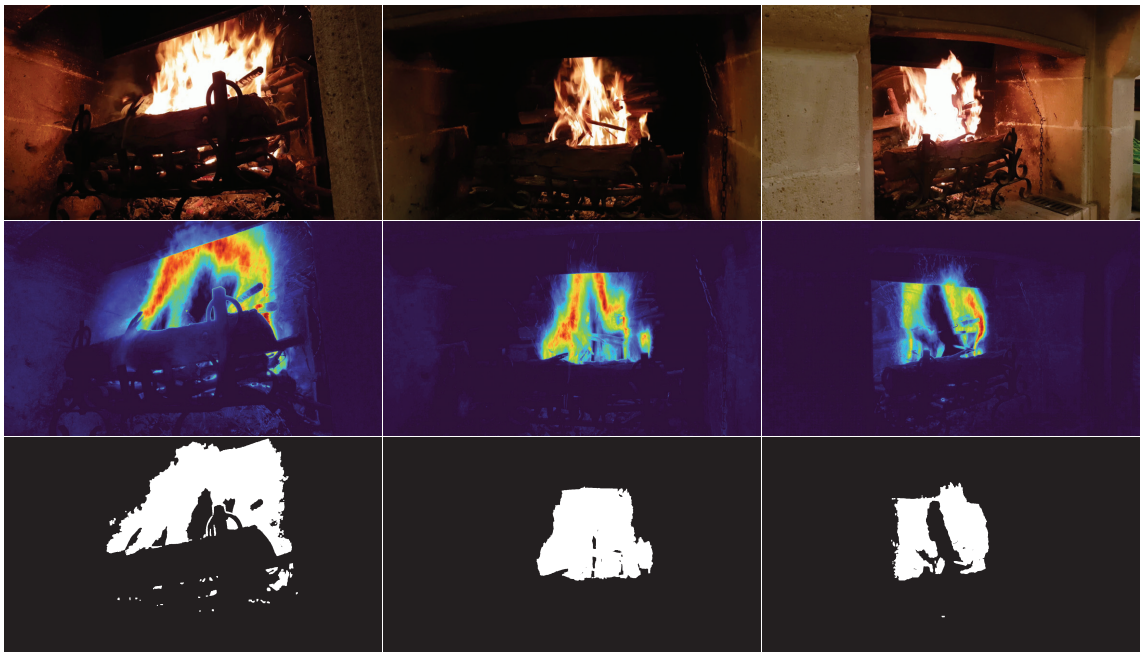


Figure 5.8: **Rough binary masks for dynamism using total variation**. Three frames from input videos (top), with their corresponding per-pixel total variation (middle), and their corresponding binary masks obtained by thresholding the total variation. Since there is only one mask per video it shows roughly what are the pixels which are dynamic at some point in the video.

5.5.1 3D localization

5.5.1.1 Dynamic video regions extraction

We start by identifying the regions of the input videos that actually capture a dynamic effect. We first extract a per video binary mask M_b by thresholding per-pixel total variation (i.e., the sum over all frames of the L_1 distance between the color at t and $t+1$). These masks are a rough estimation of the dynamic pixels in our videos (Figure 5.8). Our goal is to first extract a volumetric representation of the effect which is consistent across views. We then propagate this depth information to the view-specific dynamic part; no matching across views is possible in this region.

5.5.1.2 Voxelized visual hull

Given the dynamic segmentation masks M_b, we can combine the information from all input videos to locate the motion in world space. Laurentini [66] introduced the concept of visual hull, which uses multi-view silhouette information to recover an approximate

Figure 5.9: **Geometric representation.** Occupancy grid for the fire scene (in green), with per view geometry (in grey) associated to the view on the bottom left.

3D shape. We compute a visual hull of our dynamic effects using a voxel grid. For each voxel we randomly sample points in its volume, and reproject them onto the input video segmentation masks. We perform visibility tests using the static geometry, assuming the static parts are opaque. We consider that a voxel belongs to the visual hull if for most of the samples, all the input videos seeing the associated point agree on its the dynamic nature. More specifically, a voxel has to be visible from at least 5 views, and its reprojection must be part of the dynamic mask in at least 90% of the input views.

5.5.1.3 Per view geometry

The voxel occupancy grid approximately localizes the dynamic effect in 3D space. During rendering, we need to reproject the textures onto a surface. Since the videos are not synchronized, there is not enough multi-view consistency information to create a single global geometry. Such geometry could also pose issues with visibility. We need a geometry which represents the video content without too much distortion and which can provide some sense of consistency when switching from one video to another. Thus we

model the dynamic effects using per-view geometry, which can be seen as approximate view-dependent "billboards" (see Figure 5.9).

Given an input view and the voxel occupancy grid, we cast a ray for each pixel and gather the depths associated to the non empty cells traversed by the ray. Since we want to assign a single depth, which is likely to be consistent with the other views, we extract the median of the per-pixel smoothed inverse depth histogram. Then we fill holes and remove outliers using morphological operations.

However, this step only provides depth information to the dynamic regions that are consistent with other views. To assign depth to the remaining dynamic regions which are specific to each video, we propagate the previously obtained foreground depth using the input video total variation as guide. More specifically, we solve the following least-squares system, with a data term E_d and a smoothness term E_s:

$$E(p, d_p) = E_d(p, d_p) + \sum_{q \in \mathcal{N}_p} E_s(p, q) \tag{5.7a}$$

$$E_d(p) = \begin{cases} \lambda_{\text{fg}} \left(d_p - d_p^{\text{fg}}\right)^2 & \text{if } d_p^{\text{fg}} \text{ is available} \\ \lambda_{\text{bg}} \left(d_p - d_p^{\text{bg}}\right)^2 & \text{otherwise} \end{cases} \tag{5.7b}$$

$$E_s(p, q) = \lambda_s \left(TV_p \cdot TV_q\right)^2 \tag{5.7c}$$

where d_p is the unknown per-pixel depth, d_p^{fg} is the foreground depth obtained from the voxel grid if available, d_p^{bg} is the background depth obtained from the static part of the MVS reconstruction, TV_p is the per-pixel RGB total variation L_1 norm, and $\mathcal{N}_p$ is the 4 neighborhood in image space, discarding neighbors across foreground occlusion edges. We use $\lambda_{\text{fg}} = 100, \lambda_{\text{bg}} = 1, \lambda_s = 0.01$ in all our experiments.

To reduce texture distortion during reprojection, we favor billboard-like geometry by smoothing the resulting depth maps in image space using a large kernel (100 pixels in radius) in an occlusion-aware fashion. Finally, we extract a per-view mesh from the depth map.

5.5.2 Video matting

The dynamic regions in the scenes such as water and flames are often semi-transparent. In such cases, the color of a pixel in the dynamic region can be modeled as a mixture of

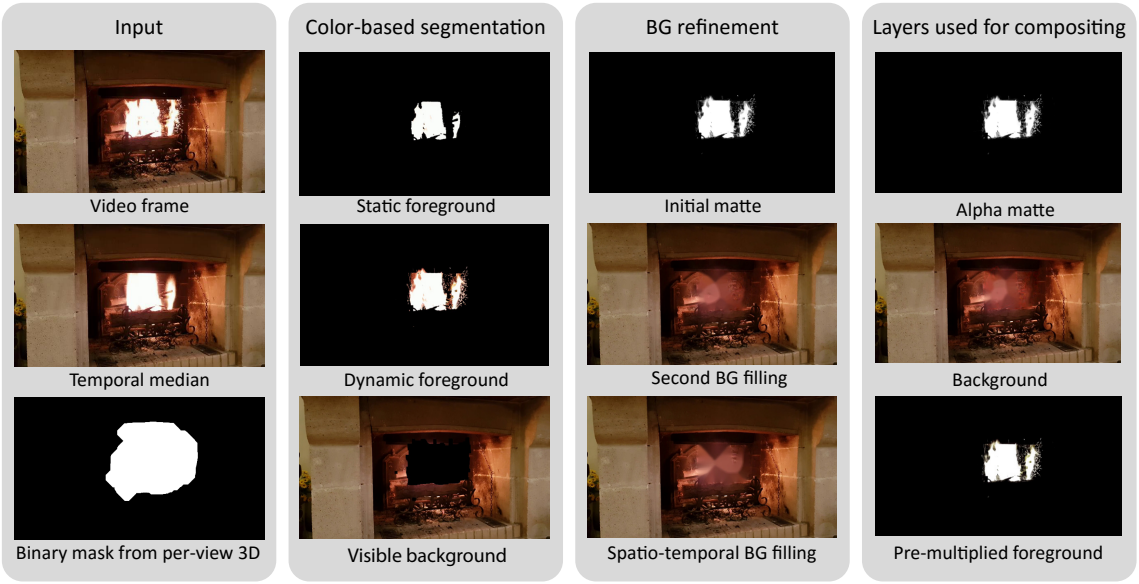


Figure 5.10: **Matting pipeline.** We start from an input video, its median frame, a binary mask corresponding to the previously computed per-view geometry and a user-provided foreground color. The goal is to extract foreground and background videos as well as the associated alpha matting video.

the static background color and the dynamic object color using the compositing equation commonly used in natural image matting. If we consider the pixel p in a single frame from a single viewpoint, the compositing equation is written as:

$$\mathbf{c}_p = \alpha_p \mathbf{f}_p + (1 - \alpha)\mathbf{b}_p, \tag{5.8}$$

where α_p is the opacity of the dynamic foreground, and $\mathbf{c}_p$, $\mathbf{f}_p$ and $\mathbf{b}_p$ are the observed pixel, dynamic foreground, and static background colors in RGB, respectively. As α_p, $\mathbf{f}_p$, and $\mathbf{b}_p$ are all unknown in this equation, this is a highly underconstrained problem. This is typically solved using significant user intervention in the natural matting literature.

Instead, we simplify the problem by taking advantage of our particular setup. One assumption we make is about $\mathbf{f}_p$. Since the foreground objects we consider are quite uniform in color, we use a single $\mathbf{f}$ in the initial steps of our method. Also, given the static camera setup, we exploit the temporal information on the background to reliably determine $\mathbf{b}_p$ in a majority of the pixels. This way, we are able to obtain a clear opacity estimation with very little user input. We give the step-by-step explanation of our matting pipeline in the rest of this section.

An overview of our matting method is shown in Figure 5.10. Our matting pipeline requires a foreground color $\mathbf{f}$ as the only user input. This color is determined once for each dataset. We use the binary mask which marks the dynamic regions from the 3-D reconstruction as described in the previous section, and proceed with opacity estimations for each viewpoint independently.

We first start by determining the fully foreground (i.e. opaque) and dynamic foreground regions in all the frames of a viewpoint using color-based segmentation. The dynamic foreground mask represents the regions where the background is visible in some of the frames behind the dynamic objects. We reason about the dynamic and static foreground regions by using the temporal median frame, i.e., the median color of each pixel in the temporal dimension, together with the input frame. We mark the pixels that have a very similar color to $\mathbf{f}$ in both the median and the current frame as static foreground and reason that since α_p is likely unity in these regions, then estimating $\mathbf{b}_p$ is of little importance. The dynamic foreground regions, determined by good color matches to either median or current frame, are used as candidate regions for temporal inpainting of the background. By analyzing these regions in each frame, for each pixel p, we determine the frames where the background color $\mathbf{b}_p$ is fully visible and construct a background frame.

In this setup, $\mathbf{b}_p$ is a determining factor for a reliable matte estimation. Therefore, we refine our temporal inpainting of the background in a second step. First we fill in the pixels where the background was never fully observed by a simple nearest-neighbor inpainting. We then use this intermediate background image for opacity estimation by solving Equation 5.8 for α_p. We repeat the temporal inpainting of the background using this refined map by filling in $\mathbf{b}_p$ when $\alpha_p = 0$ in a frame. This more complete background image is then ready for the final opacity estimation.

We determine the per-pixel per-frame opacities using this refined static background image. While we have modeled the foreground as a single color until now, for a realistic rendering of the dynamic regions, we would like to determine the subtle color variations around the semi-transparent regions. For this purpose, we use a modified version of the information-flow layer color estimation (IFL) method [5]. The layer color estimation methods typically aim to estimate $\mathbf{f}_p$ and $\mathbf{b}_p$ given the alpha matte and the original image. IFL defines a linear system of equations which includes energy definitions that

target spatial consistency as well as nonlocal consistency between pixels based on color similarity for better stability in this under-constrained problem. The problem is better constrained in our case since we already have a reliable estimation of the background colors. Hence, we replace the nonlocal energy terms in IFL formulation with a quadratic cost on fidelity towards the estimated background colors and optimize for the final dynamic foreground colors that are used in our rendering pipeline.

5.6 Rendering

We now have all the elements necessary for free-viewpoint navigation of scenes, including stochastic phenomena such as flames, water streams, and fountains etc., captured in a casual manner with a single video camera.

At each frame, we need to synthesize the image corresponding to a novel view. For the foreground, we choose the 4 closest input views, and render their per-view meshes. We query and then reproject the per temporal frequency RGBA textures from the foreground videos and perform ULR blending independently per frequency. All the levels are summed to compose the final foreground layer. For the background we chose the 8 closest input views to compute a background ULR using frames for the background videos. Finally we alpha-blend the foreground and the background in a final pass.

In practice, there are two additional specific cases which occur frequently. The first involves dynamic phenomena (typically secondary illumination effects) which are projected onto well-reconstructed geometry, e.g., the light cast by flames on the walls of a fireplace (Figure 5.11). The second involves the case where the dynamic phenomenon is fully transparent with two levels of depth which need to be treated separately, such as the surface of water and the sea floor.

5.6.1 Reconstructed dynamic surfaces

So far we have addressed scene motion which breaks the assumption of multi-view consistency. However, such direct sources of variation are also responsible for secondary sources of dynamic effects in a scene. A moving object can also produce changes of illumination, whether it is a source of illumination itself such as a fire, or if it reflects some portion of the incoming light, such as moving water. As a consequence, static surfaces hit by the outgoing light of the dynamic effects have time-dependent textures (Figure 5.11).

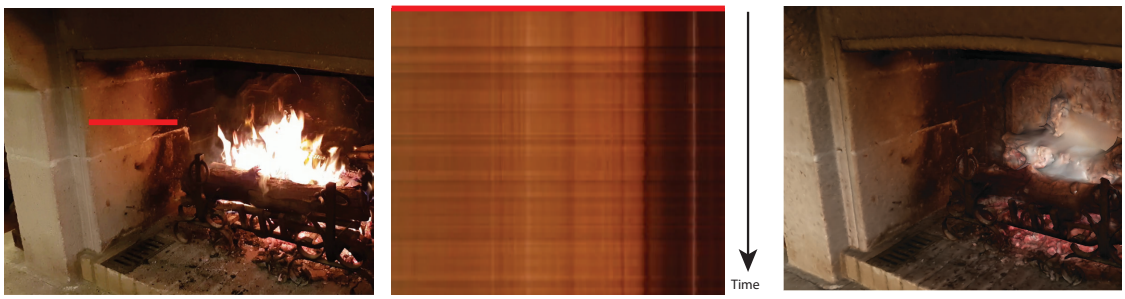


Figure 5.11: **Time-dependent background.** The dynamic flames indirectly change the appearance of the static background (horizontal patterns in the middle slice). However, they can be reprojected in novel views in a time-dependent manner onto the reconstructed geometry (colored mesh, far right).

Since these changes over time have much lower intensity, they are usually not a problem for multi-view-stereo algorithms, and underlying surfaces are reconstructed along with normal static geometry.

However, to better reproduce the overall ambiance of the captured scene, these surfaces need to be rendered using time and view-dependent textures. To do this, we use the background videos computed from subsection 5.5.2 as time-dependent textures. As their variation in time is much more subtle than the primary source of dynamism, we can simply blend them using a per-frame standard IBR technique such as ULR [14]. A standard occlusion-aware alpha compositing between the foreground and background layer can then be performed.

5.6.2 Transparent surfaces

For transparent surfaces such as clean water, the different layers of the temporal Laplacian pyramid correspond to different visual phenomena. For water, high frequencies are associated with wave displacements at the water surface, while low frequencies are associated with the transmitted colors since on temporal average, waves are canceled out. For such effects, we can extend the previous methods to not only have per-view geometries, but also per-frequency geometries. In the case of clear water, this allows distinct and correct parallax for both the waves and the sea floor. Note that this special case only works if, for any novel viewpoint, there is a visible low-frequency geometry behind the high-frequency geometry. Indeed, the temporal collapse of the renderings requires a low frequency residual, which is not guaranteed to exist in the general case if

the geometry depends on the frequency.

5.7 Results

5.7.1 Implementation

Our implementation uses C++ and OpenGL for the interactive rendering. Our method runs at $5 \sim 10$ fps on a NVidia GTX 1080, blending 4 videos with 8 frequency layers at each frame at a resolution of 800x600. For stop motion effects, where the novel view moves but videos are stopped, no texture update is done and the methods runs at 30 fps at a resolution of 1920x1080. The current bottleneck is the upload at each frame of around 40 uncompressed textures from RAM to VRAM. As most of the textures are layers from temporal Laplacian pyramids, many pixels have zero values, suggesting possible compression strategies for more efficiency.

5.7.2 View synthesis

We provide results for two scenes where the effect is semi-transparent in Figure 5.12 and Figure 5.13. The *Fire* scene contains a fire pit which is highly time-dependent and semi-transparent. The fire also produces indirect lighting on the background chimney. The combination of both effects reproduces the atmosphere of the input videos. The *Cave* scene contains two semi-transparent main waterfalls and an opaque water plane. There are also several indirect time-dependent lighting effects on the background rocks. We also show a result for a fully opaque scene in Figure 5.14. We can assume the sea is opaque in this scene since the parallax between the water surface and the sea-floor are similar when the sea-floor is visible. We also show a result for a fully transparent scene in Figure 5.15. For the *Seaside* scene, temporal frequencies are projected onto two different geometries, allowing correct parallax when moving the viewpoint. The highest frequencies are on the water plane, while the low frequencies show the sea-floor as waves are canceled out. We show that by selecting a subset of the available frequencies, we have some control over the sea "liveliness" in Figure 5.16. Such an editing can be done at runtime, by simply rendering only a subset of the foreground pyramid layers.

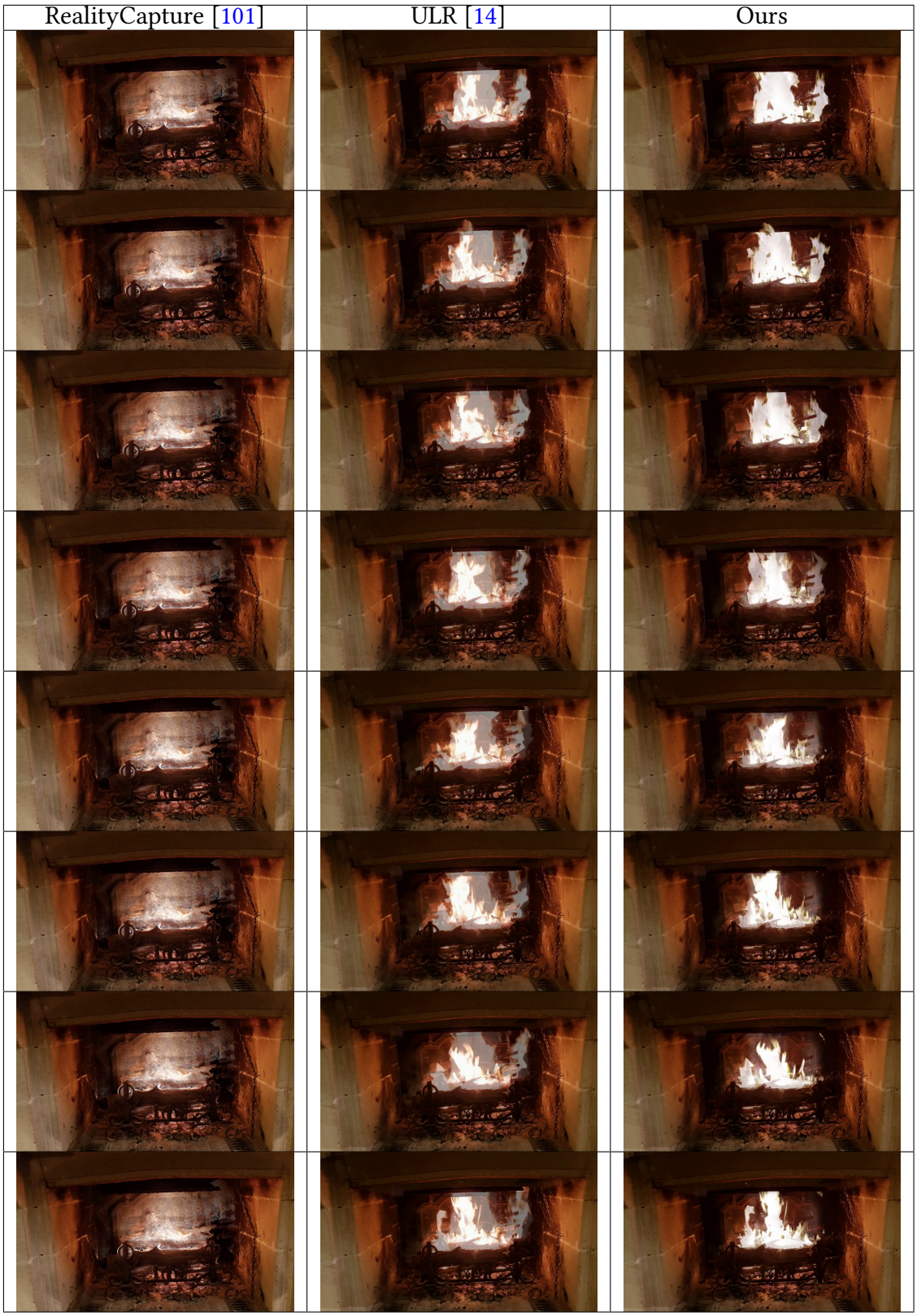


Figure 5.12: Results for the *Fire* scene with consecutive frames. From left to right: rendering using RealityCapture [101], our per-frame improved version of ULR [14], and our result. Our blending reduces the ghosting artifacts when blending unsynchronized content.

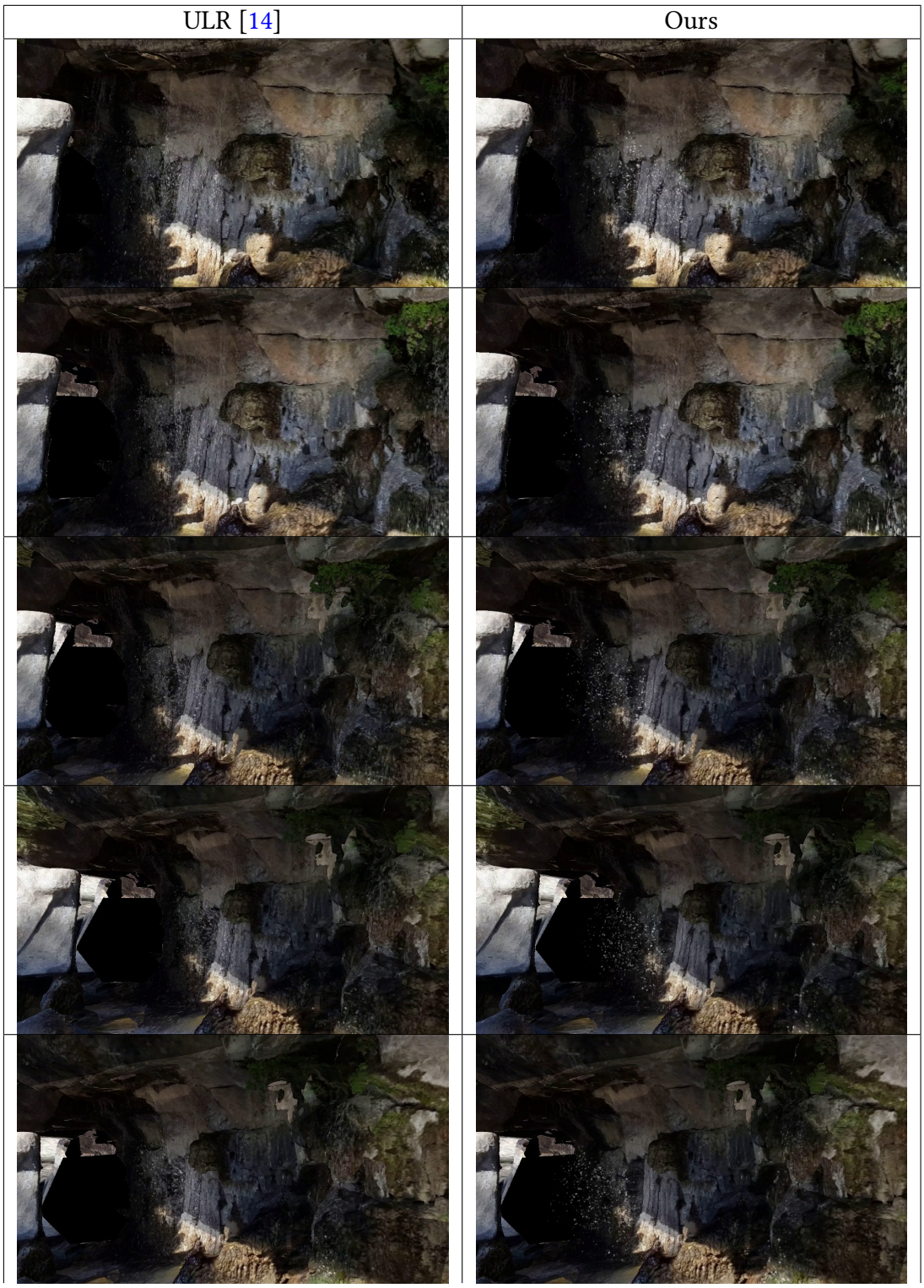


Figure 5.13: Results for the *Cave* scene. From left to right: rendering using RealityCapture [101], our per-frame improved version of ULR [14], and our result. Results are best seen when zoomed in on a pdf viewer.

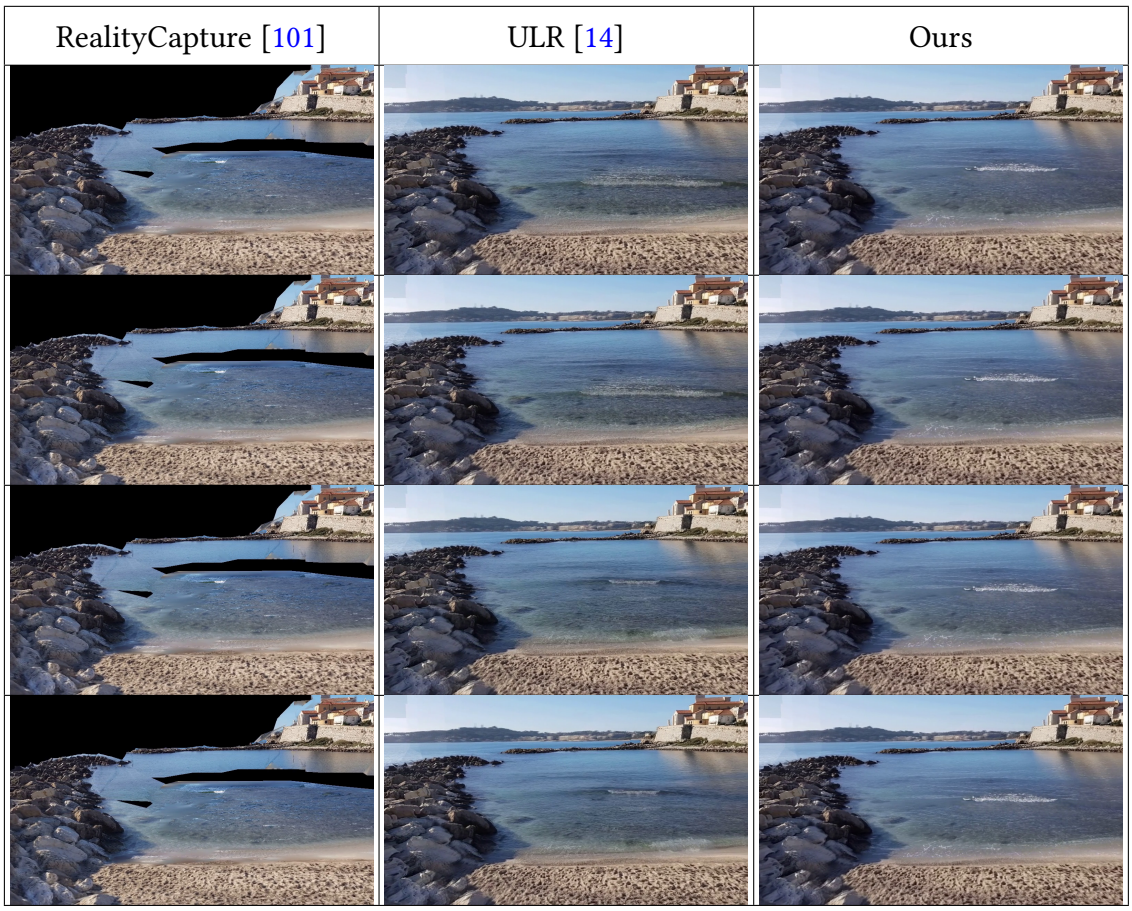


Figure 5.14: Results for the *Beach* scene for consecutive frames. From left to right: rendering using RealityCapture [101], our per-frame improved version of ULR [14] and our result. Notice the non-looping video transition on the middle column between the second and third row, and the more blurred result of ULR.

5.8 Conclusion

We provided a solution to the problem of stochastic time-dependent effects, which allows video based rendering to be used while keeping a similar level of complexity as traditional image based rendering. Our method provides both the casual capture and the free-viewpoint rendering of traditional wide-baseline IBR. We extended an image pyramid based decomposition to the time dimension to have video representations which allows seamless video looping and multi-view video blending for challenging scenes. Such a decomposition also allows some level of scene editing. By selecting which temporal frequency is used at runtime, the user can modify the degree of "liveliness" in the

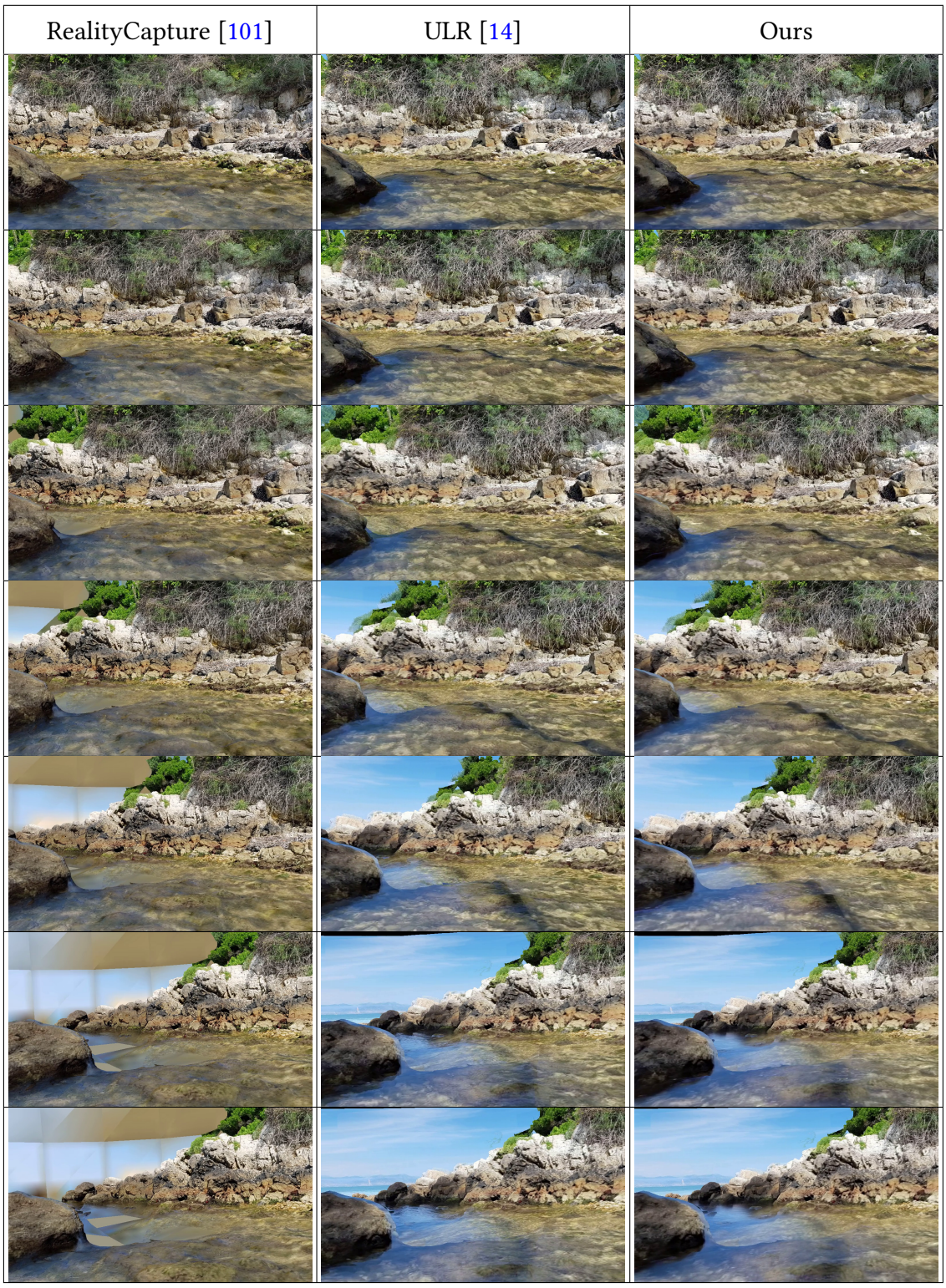


Figure 5.15: Results for the *Seaside*. From left to right: rendering using RealityCapture [101], our per-frame improved version of ULR [14], and our result. The low frequencies are reprojected onto the sea-floor while the high frequencies are reprojected onto the water plane, allowing correct parallax when moving the viewpoint. Results are best seen when zoomed in on a pdf viewer.

scene.

In conclusion, we proposed a method which provides a natural extension of IBR to some type of dynamic scenes, overcoming the time-dependency and un-synchronized capture challenges. In this regard, we demonstrated the possibility of "bringing IBR scenes to life".

For future work, we aim to better model the volumetric geometry of the dynamic effects, which would allow better transitions between viewpoints. Indeed, the current median depth approach provides plausible consistency at the middle of the effect, but the transition quality decreases near the per-view geometry boundaries. Moreover, the current visual hull approach is a limitation of the capture setup, forcing the user to have a wide angular coverage of the effect.

Figure 5.16: **Scene editability**. By partially collapsing the temporal Laplacian pyramid, we can chose the frequencies to keep in the input videos. In the *Seaside* scene, it is possible to control the sea agitation. Keeping the notation from Equation 5.3, we show for video i a frame of the partially collapsed video $\mathcal{K}[V_i]_\ell + \mathcal{K}[V_i]_{\ell+1} + \cdots + \mathcal{K}[V_i]_{n_\mathrm{L}-1}$, for $\ell = 0$ (top left) to $\ell = 5$ (bottom right).

Chapter 6

Conclusion

Contents

In this final chapter we review the contributions made in this thesis. Following our goal to improve the versatility of Image Based Rendering, we review the research impact of our results and discuss possible research directions to overcome the remaining limitations of IBR.

6.1 Contributions

IBR is be a powerful approach to capture and render realistic environments at low cost, both in terms of capture setup and computational power required for rendering. However, images are only screenshot of reality at a given time and make the captured scene hard to modify. To address this problem, we proposed a multi-view consistent inpainting algorithm which enables editing of IBR scenes in a free-viewpoint setting. Our method allows the removal of objects from an IBR scene, either from manually created binary masks, or from 2D bounding boxes found automatically by an object detection network. These removals provide an increase in rendering quality for scenes containing objects which are difficult to reconstruct via MVS. They also allow the movement of objects which are well reconstructed, enabling some scene editing by displacement. We demonstrated our algorithm on several urban scenes.

One major drawback of free-viewpoint IBR is the restricted classes of scenes different algorithms and treat. Indeed, because of the photo-consistency hypothesis, many scenes suffer from rendering artifacts or cannot be treated at all, such as scenes containing thin structures, e.g., fences or railings. We have proposed a method to deal with the problem of such thin structures for image-based rendering. Our approach is a multi-layered segmentation algorithm which exploits multi-view information and 3D reconstruction to provide segmentation of thin structures supported by surfaces with simple geometry. Our semi-automatic solution produces results showing improvement over previous methods in terms of rendering quality, allowing common scenes with thin structures to be used in a free-viewpoint IBR context.

Another limitation of IBR is its static nature, turning captured scenes into “still life”. We demonstrated that, in the case of stochastic texture, we can preserve casual capture setup and free-viewpoint navigation while “bringing to life” scenes by using additional videos to the standard IBR input. We proposed a representation for videos which enables natural video looping and seamless multi-view video spatio-temporal blending.

6.2 Research impact

The research projects we presented in this thesis led to presentations in national and international conferences. Results from the inpainting project from chapter 3 were presented at the Conference on 3D Vision (3DV) 2016 (poster) and at the j•Fig (Computer Graphic French Association) 2016 (oral). Results from the thin structures project from chapter 4 were presented at EGSR (Eurographics Symposium on Rendering) 2018 (oral). Moreover, the multi-view video project from chapter 5 is in preparation for a submission.

Our multi-view inpainting solution was part of the European project CR-PLAY[1] which aims to reduce the time and expense involved in creating videogame assets. We contributed to making high quality realistic content accessible to small game developers. Therefore these initial ideas will potentially have an impact on the development of future *indie* games. The thin structure search was part of the French National Research Agency project SEMAPOLIS[2], whose goal is to produce large-scale image analysis and semantized 3D reconstructions for urban data visualization. As thin structure are omnipresent in human made environments and are a challenge for both segmentation and rendering, we contributed to improve the visual quality of such urban visualizations. Finally, the research presented on multi-view videos is part of the ERC Advanced grant project FUNGRAPH[3], which aims to unify the use of both approximate captured content and perfect hand-modeled content in rendering. The goal is to develop a framework to model input data uncertainty in order to guide the compositing. Our imperfect geometric representation for dynamic effects and its impact on the rendering quality provide initial insights for further generalizations.

At the scale of our research group, the work presented in this thesis had an impact on other projects. chapter 3 served as basis for the method proposed by Philip and Drettakis [97]. chapter 4 inspired and was inspired by the work of Rodriguez et al. [102], both in terms of code resources and ideas.

[1] http://www.cr-play.eu/
[2] https://project.inria.fr/semapolis/
[3] http://fungraph.inria.fr

6.3 Future work

Semantic information. We used superficially semantic information to guide our algorithms. In the chapter 3, we used semantic information to extract the classes of objects that can be problematic for IBR rendering. We also used the geometry normals as basic semantic information to extract priors on where to copy the created content from. For chapter 4, we used the assumption that thin structures could be mostly supported by surfaces with simple geometry.

However, progress in computer vision and especially in deep neural networks could make the extraction and usage of semantic information more frequent. Per-pixel semantic information could be used to help identify potential sources of rendering artifacts such as reflectives surfaces or vegetation. Such a scheme could generalize both the inpainting and thin structures project as one could automatically determine if regions of the input images should be removed or, on the contrary, whether they should have special treatment, e.g. to refine the geometry.

Semantic information could be very useful to treat the hardest cases for IBR, for example reflective and semi-transparent surfaces. For both cases, the color information associated to a pixel cannot be assigned to a single depth on the input ray. For reflective surfaces, it corresponds to the reflected content, and for transparent surfaces, it correspond to an accumulation of color from multiple depths. If these regions are properly detected, one could imagine that rendering algorithms could be designed to correctly handle such phenomena.

Metric for rendering quality. Evaluating the quality of IBR methods objectively is a long standing open problem. As IBR algorithms aim to produce plausible novel views, there is a theoretical ground truth from the associated camera. However, any deviation from this correct output is purely determined by user perception, which is not objective. In this thesis, we were not exposed to this problem as previous methods had inherent difficulties handling our scenes. Therefore, our comparisons did not suffer from the lack of objectivity as we had to compare with renderings with strong visible artifacts. Yet, this lack of quantitative evaluation remains unsatisfactory.

Previous methods used the leave one out metric [90, 53], using the input images as sparse sources of ground-truth. However, as the rendering artifacts are mostly localized in the

output, global image metrics such as sum of square differences provide no useful information for evaluation. For example, artifacts due to incorrect occlusion handling are one of the most visible perceptually, yet they are only located on strong image edges, which are very sparse in terms of the number of output pixels. Moreover, since evaluation can be performed only on a discrete set of input views, temporal consistency is impossible to check.

Recent IBR methods [52, 115] address this issue by using perceptual losses, based on the latent image vectors reused from classification deep neural networks. However, they remain limited to image-to-image comparison. While developing for better image metric is a vast research topic on its own, it would be interesting to explore metrics which are more specific to IBR setups. Especially, evaluating temporal consistency in a general way when moving the viewpoint is promising for rendering based optimizations.

Research has also been completed about human visual attention in images and videos in a more general context [68]. Methods evaluate visual quality in specific scenarios such as videos [87], image stereo [12] or in Depth Image Based Rendering [103]. They propose both subjective and objective metrics that have high correlation with human judgment. However, they are yet to be applied to wide-baseline free-viewpoint IBR setups and their complexity might be an obstacle for machine learning applications.

Dynamic behavior. As we showed in chapter 5, capturing dynamic behavior in a scene is important for user immersion. However, as many static scenes are already challenging, adding an additional dimension to the problem makes it severely under-constrained and possibly intractable. Current methods either put constraints on the capture setup [46], on the user navigation [8, 124], or, as the method we presented in this thesis, on the type of handled motion [19, 25].

Research remains to be completed to evaluate how much dynamic content needs to be extracted to preserve the feeling the user wanted to capture. Identifying more precisely the semantics of the effect and the level of quality required for plausible rendering could lead to the capability of capturing more types of motion in a casual manner.

Virtual Reality. With the emergence of Virtual Reality (VR) devices such as the HTC Vive or Oculus Rift, exploring virtual environments with plausible immersion is possible for the casual user. Virtual reality can be used in entertainment applications such as

video gaming or movies, but also for educational, training, or medical purposes. As the level or realism of the virtual environment is crucial for the user immersion, image and video based rendering have a promising future in VR, thanks to their easy-to-use content creation and rendering capabilities. Moreover, free-viewpoint navigation IBR provides compelling position and angular freedom where most head-mounted displays (HMD) setups currently allow only head rotation and small displacements.

However many challenges have to be solved before IBR can be used widely for VR. HMDs require high frame rates to not expose users to *virtual reality sickness*. Most IBR methods have difficulties providing high quality with both high resolution and very high framerates. The issue is even more complex as IBR algorithms often use different tools from the traditional rendering pipeline, for which the HMDs hardware and software might be not adapted.

One final challenge which must be solved to achieve real immersion is to capture and to render high dynamic range images (HDR). Using such enhanced data would also allow better editing capabilities and more realistic tone mapping. However, dealing with HDR images requires noticeably more computing capabilities which is a scarce resource in HMD applications.

To conclude, the results in this thesis have significantly advanced the versatility and image quality for image and video based rendering, but there is still much to be completed before such techniques can see widespread use across many application areas.

Bibliography

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. "SLIC superpixels compared to state-of-the-art superpixel methods". In: *IEEE transactions on pattern analysis and machine intelligence* 34.11 (2012), pp. 2274–2282 (Cited on page 20).

[2] Edward H Adelson and James R Bergen. "The plenoptic function and the elements of early vision". In: (1991) (Cited on page 10).

[3] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. "Interactive digital photomontage". In: *ACM Transactions on Graphics (ToG)*. Vol. 23. 3. ACM. 2004, pp. 294–302 (Cited on page 29).

[4] Aseem Agarwala, Ke Colin Zheng, Chris Pal, Maneesh Agrawala, Michael Cohen, Brian Curless, David Salesin, and Richard Szeliski. "Panoramic video textures". In: *ACM Transactions on Graphics (TOG)* 24.3 (2005), pp. 821–827 (Cited on page 82).

[5] Yagiz Aksoy, Tunc Ozan Aydin, and Marc Pollefeys. "Designing effective inter-pixel information flow for natural image matting". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 29–37 (Cited on page 98).

[6] Seung-Hwan Baek, Inchang Choi, and Min H Kim. "Multiview image completion with space structure propagation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 488–496 (Cited on pages 30, 44–46).

[7] Jiamin Bai, Aseem Agarwala, Maneesh Agrawala, and Ravi Ramamoorthi. "Selectively de-animating video." In: *ACM Trans. Graph.* 31.4 (2012), pp. 66–1 (Cited on pages 82, 89).

[8] Luca Ballan, Gabriel J Brostow, Jens Puwein, and Marc Pollefeys. "Unstructured video-based rendering: Interactive exploration of casually captured videos". In: *ACM Transactions on Graphics (TOG)*. Vol. 29. 4. ACM. 2010, p. 87 (Cited on pages 80, 82, 113).

[9] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. "PatchMatch: A randomized correspondence algorithm for structural image editing". In: *ACM Transactions on Graphics (ToG)*. Vol. 28. 3. ACM. 2009, p. 24 (Cited on pages 28, 32).

[10] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features". In: *European conference on computer vision*. Springer. 2006, pp. 404–417 (Cited on page 14).

[11] Pravin Bhat, C Lawrence Zitnick, Noah Snavely, Aseem Agarwala, Maneesh Agrawala, Michael Cohen, Brian Curless, and Sing Bing Kang. "Using photographs to enhance videos of a static scene". In: *Proceedings of the 18th Eurographics conference on Rendering Techniques*. Eurographics Association. 2007, pp. 327–338 (Cited on page 29).

[12] Emilie Bosc, Romuald Pepion, Patrick Le Callet, Martin Koppel, Patrick Ndjiki-Nya, Muriel Pressigout, and Luce Morin. "Towards a new quality metric for 3-D synthesized view assessment". In: *IEEE Journal of Selected Topics in Signal Processing* 5.7 (2011), pp. 1332–1343 (Cited on page 113).

[13] Yuri Boykov, Olga Veksler, and Ramin Zabih. "Fast approximate energy minimization via graph cuts". In: *IEEE Trans. PAMI* 23.11 (2001), pp. 1222–1239 (Cited on page 58).

[14] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. "Unstructured lumigraph rendering". In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM. 2001, pp. 425–432 (Cited on pages 16, 17, 35, 58, 69–71, 73, 81, 88, 89, 92, 100, 102–105).

[15] Peter J Burt and Edward H Adelson. "A multiresolution spline with application to image mosaics". In: *ACM transactions on Graphics* 2.4 (1983), pp. 217–236 (Cited on pages 85, 86, 88, 90).

[16] Neill DF Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. "Automatic object segmentation from calibrated images". In: *Visual Media Production (CVMP), 2011 Conference for*. IEEE. 2011, pp. 126–137 (Cited on page 54).

[17] John Canny. "A computational approach to edge detection". In: *IEEE Trans. PAMI* 6 (1986), pp. 679–698 (Cited on page 67).

[18] Owen Carmichael and Martial Hebert. "Shape-based recognition of wiry objects". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.12 (2004), pp. 1537–1552 (Cited on page 55).

[19] Joel Carranza, Christian Theobalt, Marcus A Magnor, and Hans-Peter Seidel. *Free-viewpoint video of human actors*. Vol. 22. 3. ACM, 2003 (Cited on pages 80, 81, 113).

[20] Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. "Plenoptic sampling". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 2000, pp. 307–318 (Cited on page 13).

[21] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. "Depth synthesis and local warps for plausible image-based navigation". In: *ACM Transactions on Graphics (TOG)* 32.3 (2013), p. 30 (Cited on pages 19, 20, 28, 32, 34, 35, 41, 43, 47, 49, 52, 55, 73, 81).

[22] Gaurav Chaurasia, Olga Sorkine, and George Drettakis. "Silhouette-Aware Warping for Image-Based Rendering". In: *Computer Graphics Forum*. Vol. 30. 4. Wiley Online Library. 2011, pp. 1223–1232 (Cited on page 20).

[23] Shenchang Eric Chen. "Quicktime VR: An image-based approach to virtual environment navigation". In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM. 1995, pp. 29–38 (Cited on pages 12, 14).

[24] Shenchang Eric Chen and Lance Williams. "View interpolation for image synthesis". In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. ACM. 1993, pp. 279–288 (Cited on page 11).

[25] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. "High-quality streamable free-viewpoint video". In: *ACM Transactions on Graphics (ToG)* 34.4 (2015), p. 69 (Cited on pages 80, 81, 113).

[26] Antonio Criminisi, Patrick Perez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting". In: *IEEE Trans. on image processing* 13.9 (2004), pp. 1200–1212 (Cited on page 54).

[27] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting". In: *IEEE Transactions on image processing* 13.9 (2004), pp. 1200–1212 (Cited on pages 28, 37).

[28] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B Goldman, and Pradeep Sen. "Image melding: Combining inconsistent images using patch-based synthesis." In: *ACM Trans. Graph.* 31.4 (2012), pp. 82–1 (Cited on pages 29, 41).

[29] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-based Approach". In: SIGGRAPH '96. ACM, 1996 (Cited on pages 12–14).

[30] Paul Debevec, Yizhou Yu, and George Borshukov. "Efficient view-dependent image-based rendering with projective texture-mapping". In: *Rendering Techniques' 98*. Springer, 1998, pp. 105–116 (Cited on page 13).

[31] Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. "Single-image svbrdf capture with a rendering-aware deep network". In: *ACM Transactions on Graphics (TOG)* 37.4 (2018), p. 128 (Cited on page 4).

[32] Abdelaziz Djelouah, Jean-Sébastien Franco, Edmond Boyer, Francois Le Clerc, and Patrick Pérez. "Multi-view object segmentation in space and time". In: *ICCV*. 2013, pp. 2640–2647 (Cited on page 54).

[33] Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson De Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. "Floating textures". In: *Computer graphics forum*. Vol. 27. 2. Wiley Online Library. 2008, pp. 409–418 (Cited on pages 18, 69).

[34] Cass Everitt. "Interactive Order-Independent Transparency". In: (2001). URL: http://www.nvidia.com/attach/6545 (Cited on page 69).

[35] Andrew Fitzgibbon, Yonatan Wexler, and Andrew Zisserman. "Image-based rendering using image-based priors". In: *International Journal of Computer Vision* 63.2 (2005), pp. 141–151 (Cited on pages 23, 29).

[36] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. "Deepstereo: Learning to predict new views from the world's imagery". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5515–5524 (Cited on page 24).

[37] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. "Manhattan-world stereo". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 1422–1429 (Cited on page 18).

[38] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. "Towards internet-scale multi-view stereo". In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 1434–1441 (Cited on page 16).

[39] Yasutaka Furukawa and Jean Ponce. "Accurate camera calibration from multi-view stereo and bundle adjustment". In: *CVPR*. IEEE. 2008, pp. 1–8 (Cited on page 55).

[40] David Gallup, Jan-Michael Frahm, and Marc Pollefeys. "Piecewise planar and non-planar stereo for urban scene reconstruction". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 1418–1425 (Cited on page 19).

[41] Spyros Gidaris and Nikos Komodakis. "Object detection via a multi-region and semantic segmentation-aware cnn model". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1134–1142 (Cited on page 42).

[42] Michael Goesele, Jens Ackermann, Simon Fuhrmann, Carsten Haubold, Ronny Klowsky, Drew Steedly, and Richard Szeliski. "Ambient point clouds for view interpolation". In: *ACM Transactions on Graphics (TOG)*. Vol. 29. 4. ACM. 2010, p. 95 (Cited on page 18).

[43] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. "Multi-view stereo for community photo collections". In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8 (Cited on pages 16, 55, 80).

[44] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. "The lumigraph". In: *Siggraph*. Vol. 96. 30. 1996, pp. 43–54 (Cited on pages 11, 12, 14).

[45] Miguel Granados, Kwang In Kim, James Tompkin, Jan Kautz, and Christian Theobalt. "Background inpainting for videos with dynamic objects and a free-moving camera". In: *European Conference on Computer Vision*. Springer. 2012, pp. 682–695 (Cited on page 29).

[46] James Gregson, Michael Krimerman, Matthias B Hullin, and Wolfgang Heidrich. "Stochastic tomography and its applications in 3D imaging of mixing fluids." In: *ACM Trans. Graph.* 31.4 (2012), pp. 52–1 (Cited on pages 83, 93, 113).

[47] James Hays and Alexei A Efros. "Scene completion using millions of photographs". In: *ACM Transactions on Graphics (TOG)* 26.3 (2007), p. 4 (Cited on page 30).

[48] James Hays, Marius Leordeanu, Alexei A Efros, and Yanxi Liu. "Discovering texture regularity as a higher-order correspondence problem". In: *ECCV*. Springer. 2006, pp. 522–535 (Cited on pages 53, 54).

[49] Kaiming He, Christoph Rhemann, Carsten Rother, Xiaoou Tang, and Jian Sun. "A global sampling method for alpha matting". In: *CVPR*. IEEE. 2011, pp. 2049–2056 (Cited on pages 58, 68).

[50] Mingming He, Jing Liao, Pedro V Sander, and Hugues Hoppe. "Gigapixel Panorama Video Loops". In: *ACM Transactions on Graphics (TOG)* 37.1 (2017), p. 3 (Cited on page 82).

[51] Li-wei He, Jonathan Shade, Steven Gortler, and Richard Szeliski. "Layered depth images". In: (1998) (Cited on pages 13, 14, 55).

[52] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. "Deep Blending for Free-Viewpoint Image-Based Rendering". In: *ACM Transactions on Graphics* 37.6 (2017) (Cited on pages 24, 80, 81, 113).

[53] Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. "Scalable inside-out image-based rendering". In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), p. 231 (Cited on pages 20, 24, 81, 112).

[54] Benno Heigl, Reinhard Koch, Marc Pollefeys, Joachim Denzler, and Luc Van Gool. "Plenoptic modeling and rendering from image sequences taken by a hand-held camera". In: *Mustererkennung 1999*. Springer, 1999, pp. 94–101 (Cited on page 16).

[55] Manuel Hofer, Michael Maurer, and Horst Bischof. "Efficient 3D scene abstraction using line segments". In: *Computer Vision and Image Understanding* 157 (2017), pp. 167–178 (Cited on page 55).

[56] Joel Howard, Bryan Morse, Scott Cohen, and Brian Price. "Depth-based patch scaling for content-aware stereo image completion". In: *IEEE winter conference on applications of computer vision*. IEEE. 2014, pp. 9–16 (Cited on page 30).

[57] Jia-Bin Huang, Sing Bing Kang, Narendra Ahuja, and Johannes Kopf. "Image completion using planar structure guidance". In: *ACM Transactions on graphics (TOG)* 33.4 (2014), p. 129 (Cited on pages 28, 44, 46).

[58] Michal Jancosek and Tomás Pajdla. "Multi-view reconstruction preserving weakly-supported surfaces". In: *CVPR 2011*. IEEE. 2011, pp. 3121–3128 (Cited on pages 15, 16, 32, 55, 56).

[59] Nianjuan Jiang, Ping Tan, and Loong-Fah Cheong. "Multi-view repetitive structure detection". In: *2011 Int. Conference on Computer Vision*. IEEE. 2011, pp. 535–542 (Cited on page 54).

[60] Alexandre Kaspar, Boris Neubert, Dani Lischinski, Mark Pauly, and Johannes Kopf. "Self tuning texture optimization". In: *Computer Graphics Forum*. Vol. 34. 2. Wiley Online Library. 2015, pp. 349–359 (Cited on page 68).

[61] Felix Klose, Oliver Wang, Jean-Charles Bazin, Marcus Magnor, and Alexander Sorkine-Hornung. "Sampling based scene-space video processing". In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), p. 67 (Cited on pages 28, 29).

[62] Kalin Kolev, Thomas Brox, and Daniel Cremers. "Fast joint estimation of silhouettes and dense 3d geometry from multiple images". In: *IEEE Trans. PAMI* 34.3 (2012), pp. 493–505 (Cited on page 54).

[63] Johannes Kopf, Michael F Cohen, and Richard Szeliski. "First-person hyper-lapse videos". In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), p. 78 (Cited on page 23).

[64] Johannes Kopf, Fabian Langguth, Daniel Scharstein, Richard Szeliski, and Michael Goesele. "Image-based rendering in the gradient domain". In: *ACM Transactions on Graphics (TOG)* 32.6 (2013), p. 199 (Cited on pages 21, 52, 55).

[65] Adarsh Kowdle, Sudipta N Sinha, and Richard Szeliski. "Multiple view object cosegmentation using appearance and stereo cues". In: *ECCV*. Springer. 2012, pp. 789–803 (Cited on page 54).

[66] Aldo Laurentini. "The visual hull concept for silhouette-based image understanding". In: *IEEE Transactions on pattern analysis and machine intelligence* 16.2 (1994), pp. 150–162 (Cited on page 94).

[67] Stephane Laveau and Olivier D Faugeras. "3-D scene representation as a collection of images". In: *Proceedings of 12th International Conference on Pattern Recognition*. Vol. 1. IEEE. 1994, pp. 689–691 (Cited on page 11).

[68] Patrick Le Callet and Ernst Niebur. "Visual attention and applications in multimedia technologies". In: *Proceedings of the IEEE* 101.9 (2013), pp. 2058–2067 (Cited on page 113).

[69] Philippe Levieux, James Tompkin, and Jan Kautz. "Interactive viewpoint video textures". In: *Proceedings of the 9th European Conference on Visual Media Production*. ACM. 2012, pp. 11–17 (Cited on page 83).

[70] Marc Levoy and Pat Hanrahan. "Light field rendering". In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM. 1996, pp. 31–42 (Cited on pages 12, 14).

[71] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, et al. "The digital Michelangelo project: 3D scanning of large statues". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 2000, pp. 131–144 (Cited on page 4).

[72] Zicheng Liao, Neel Joshi, and Hugues Hoppe. "Automated video looping with progressive dynamism". In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 77 (Cited on pages 82, 89).

[73] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1925–1934 (Cited on page 77).

[74] Christian Lipski, Felix Klose, and Marcus Magnor. "Correspondence and depth-image based rendering a hybrid approach for free-viewpoint video". In: *IEEE Transactions on Circuits and Systems for Video Technology* 24.6 (2014), pp. 942–951 (Cited on pages 20, 28, 52).

[75] Dani Lischinski and Ari Rappoport. "Image-based rendering for non-diffuse synthetic scenes". In: *Rendering Techniques' 98*. Springer, 1998, pp. 301–314 (Cited on page 13).

[76] Lingjie Liu, Duygu Ceylan, Cheng Lin, Wenping Wang, and Niloy J. Mitra. "Image-based Reconstruction of Wire Art". In: *ACM SIGGRAPH 2017* (2017) (Cited on page 55).

[77] Siying Liu, Tian-Tsong Ng, Kalyan Sunkavalli, Minh N Do, Eli Shechtman, and Nathan Carr. "PatchMatch-based Automatic Lattice Detection for Near-Regular Textures". In: *ICCV*. 2015, pp. 181–189 (Cited on pages 54, 75).

[78] Y. Liu, T. Belkina, J. H. Hays, and R. Lublinerman. "Image de-fencing". In: *CVPR*. June 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587493 (Cited on page 54).

[79] Anthony Lobay and David A Forsyth. "Recovering shape and irradiance maps from rich dense texton fields". In: *CVPR*. Vol. 1. IEEE. 2004, pp. I–400 (Cited on page 53).

[80] Gerrit Lochmann, Bernhard Reinert, Tobias Ritschel, Stefan Müller, and Hans-Peter Seidel. "Real-time Reflective and Refractive Novel-view Synthesis." In: *VMV*. 2014, pp. 9–16 (Cited on page 22).

[81] David G Lowe et al. "Object recognition from local scale-invariant features." In: *iccv*. Vol. 99. 2. 1999, pp. 1150–1157 (Cited on page 14).

[82] Tobias Martin, Juan Montes, Jean-Charles Bazin, and Tiberiu Popa. "Topology-aware reconstruction of thin tubular structures". In: *SIGGRAPH Asia 2014 Technical Briefs*. ACM. 2014, p. 12 (Cited on page 55).

[83] Leonard McMillan and Gary Bishop. "Plenoptic modeling: An image-based rendering system". In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. Citeseer. 1995, pp. 39–46 (Cited on pages 11, 12).

[84] Yadong Mu, Wei Liu, and Shuicheng Yan. "Video De-fencing". In: *arXiv preprint arXiv:1210.2388* (2012) (Cited on page 54).

[85] Giljoo Nam, Joo Ho Lee, Diego Gutierrez, and Min H Kim. "Practical SVBRDF acquisition of 3D objects with unstructured flash photography". In: *ACM Transactions on Graphics (TOG)* 37.6 (2019), p. 267 (Cited on page 4).

[86] Alasdair Newson, Andrés Almansa, Matthieu Fradet, Yann Gousseau, and Patrick Pérez. "Video inpainting of complex scenes". In: *SIAM Journal on Imaging Sciences* 7.4 (2014), pp. 1993–2019 (Cited on pages 29, 34, 37, 41).

[87] Alexandre Ninassi, Olivier Le Meur, Patrick Le Callet, and Dominique Barba. "Considering temporal variations of spatial visual distortions in video quality assessment". In: *IEEE Journal of Selected Topics in Signal Processing* 3.2 (2009), pp. 253–265 (Cited on page 113).

[88] Gen Nishida, Ignacio Garcia-Dorado, Daniel G Aliaga, Bedrich Benes, and Adrien Bousseau. "Interactive sketching of urban procedural models". In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), p. 130 (Cited on page 77).

[89] Makoto Okabe, Yoshinori Dobashi, Ken Anjyo, and Rikio Onai. "Fluid volume modeling from sparse multi-view images by appearance transfer". In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), p. 93 (Cited on page 83).

[90] Rodrigo Ortiz-Cayon, Abdelaziz Djelouah, and George Drettakis. "A bayesian approach for selective image-based rendering using superpixels". In: *International Conference on 3D Vision-3DV*. 2015 (Cited on pages 20, 34, 52, 71, 73, 112).

[91] Rodrigo Ortiz-Cayon, Abdelaziz Djelouah, Francisco Massa, Mathieu Aubry, and George Drettakis. "Automatic 3d car model alignment for mixed image-based rendering". In: *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE. 2016, pp. 286–295 (Cited on pages 55, 77).

[92] Martin Ralf Oswald, Jan Stühmer, and Daniel Cremers. "Generalized connectivity constraints for spatio-temporal 3D reconstruction". In: *ECCV*. Springer. 2014, pp. 32–46 (Cited on page 55).

[93] Minwoo Park, Kyle Brocklehurst, Robert T Collins, and Yanxi Liu. "Deformed lattice detection in real-world images using mean-shift belief propagation". In: *IEEE Trans. PAMI* 31.10 (2009), pp. 1804–1816 (Cited on pages 54, 75).

[94] Minwoo Park, Kyle Brocklehurst, Robert T Collins, and Yanxi Liu. "Image defencing revisited". In: *ACCV*. Springer. 2010, pp. 422–434 (Cited on pages 52, 54, 75).

[95] Eric Penner and Li Zhang. "Soft 3D reconstruction for view synthesis". In: *ACM Transactions on Graphics (TOG)* 36.6 (2017), p. 235 (Cited on pages 20, 52, 55, 70, 71, 73, 80, 81).

[96] Patrick Pérez, Michel Gangnet, and Andrew Blake. "Poisson image editing". In: *ACM Transactions on graphics (TOG)* 22.3 (2003), pp. 313–318 (Cited on page 88).

[97] Julien Philip and George Drettakis. "Plane-based multi-view inpainting for image-based rendering in large scenes". In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM. 2018, p. 6 (Cited on pages 45, 111).

[98] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. "Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score". In: *International Journal of Computer Vision* 72.2 (2007), pp. 179–193 (Cited on page 15).

[99] Yael Pritch, Eitam Kav-Venaki, and Shmuel Peleg. "Shift-map image editing". In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 151–158 (Cited on page 29).

[100] Sergi Pujades, Frédéric Devernay, and Bastian Goldluecke. "Bayesian view synthesis and image-based rendering principles". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3906–3913 (Cited on page 23).

[101] Capturing Reality. *RealityCapture reconstruction software*. 2018. URL: https://www.capturingreality.com/Products/ (Cited on pages 16, 56, 72, 83, 102–105).

[102] Simon Rodriguez, Adrien Bousseau, Fredo Durand, and George Drettakis. "Exploiting Repetitions for Image-Based Rendering of Facades". In: *Computer Graphics Forum*. Vol. 37. 4. Wiley Online Library. 2018, pp. 119–131 (Cited on page 111).

[103] Dragana Sandić-Stanković, Dragan Kukolj, and Patrick Le Callet. "DIBR synthesized image quality assessment based on morphological wavelets". In: *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE. 2015, pp. 1–6 (Cited on page 113).

[104] Arno Schödl, Richard Szeliski, David H Salesin, and Irfan Essa. "Video textures". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 2000, pp. 489–498 (Cited on page 82).

[105] Johannes Lutz Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (Cited on pages 15, 16).

[106] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. "Pixelwise View Selection for Unstructured Multi-View Stereo". In: *European Conference on Computer Vision (ECCV)*. 2016 (Cited on page 16).

[107] Christopher Schwartz, Ralf Sarlette, Michael Weinmann, and Reinhard Klein. "DOME II: A Parallelized BTF Acquisition System." In: *Material Appearance Modeling*. 2013, pp. 25–31 (Cited on page 3).

[108] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. "A comparison and evaluation of multi-view stereo reconstruction algorithms". In: *CVPR*. Vol. 1. 2006, pp. 519–528 (Cited on page 61).

[109] Steven M Seitz and Charles R Dyer. "Physically-valid view synthesis by image interpolation". In: *Proceedings IEEE Workshop on Representation of Visual Scenes (In Conjunction with ICCV'95)*. IEEE. 1995, pp. 18–25 (Cited on page 12).

[110] Harry Shum and Sing Bing Kang. "Review of image-based rendering techniques". In: *Visual Communications and Image Processing 2000*. Vol. 4067. International Society for Optics and Photonics. 2000, pp. 2–14 (Cited on page 10).

[111] Heung-Yeung Shum, Shing-Chow Chan, and Sing Bing Kang. *Image-based rendering*. Springer Science & Business Media, 2008 (Cited on page 10).

[112] Sudipta N Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. "Image-based rendering for scenes with reflections." In: *ACM Trans. Graph.* (2012) (Cited on pages 52, 55).

[113] Sudipta N Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. "Image-based rendering for scenes with reflections." In: *ACM Trans. Graph.* 31.4 (2012), pp. 100–1 (Cited on page 21).

[114] Sudipta Sinha, Drew Steedly, and Rick Szeliski. "Piecewise planar stereo for image-based rendering". In: (2009) (Cited on page 19).

[115] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. "DeepVoxels: Learning Persistent 3D Feature Embeddings". In: *CoRR* abs/1812.01024 (2018). arXiv: 1812.01024. URL: http://arxiv.org/abs/1812.01024 (Cited on pages 24, 113).

[116] Jan Smisek, Michal Jancosek, and Tomas Pajdla. "3D with Kinect". In: *Consumer depth cameras for computer vision*. Springer, 2013, pp. 3–25 (Cited on page 4).

[117] Noah Snavely, Steven M Seitz, and Richard Szeliski. "Photo tourism: exploring photo collections in 3D". In: *ACM transactions on graphics (TOG)*. Vol. 25. 3. ACM. 2006, pp. 835–846 (Cited on pages 14, 80).

[118] Jonathan Starck and Adrian Hilton. "Surface capture for performance-based animation". In: *IEEE computer graphics and applications* 27.3 (2007), pp. 21–31 (Cited on page 81).

[119] Richard Szeliski and Heung-Yeung Shum. "Creating full view panoramic image mosaics and environment maps". In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 1997, pp. 251–258 (Cited on page 12).

[120] Justus Thies, Michael Zollhöfer, Christian Theobalt, Marc Stamminger, and Matthias Nießner. "IGNOR: Image-guided Neural Object Rendering". In: *CoRR* abs/1811.10720 (2018). arXiv: 1811.10720. URL: http://arxiv.org/abs/1811.10720 (Cited on page 24).

[121] Theo Thonat, Yağiz Aksoy, Sylvain Paris, Miika Aittala, Fredo Durand, and George Drettakis. "Practical Video-Based Rendering of Dynamic Stationary Environments from Unsynchronized Inputs". In preparation. 2019 (Cited on page 8).

[122] Theo Thonat, Abdelaziz Djelouah, Fredo Durand, and George Drettakis. "Thin structures in image based rendering". In: *Computer Graphics Forum*. Vol. 37. 4. Wiley Online Library. 2018, pp. 107–118 (Cited on page 8).

[123] Theo Thonat, Eli Shechtman, Sylvain Paris, and George Drettakis. "Multi-view inpainting for image-based scene editing and rendering". In: *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE. 2016, pp. 351–359 (Cited on page 8).

[124] James Tompkin, Kwang In Kim, Jan Kautz, and Christian Theobalt. "Videoscapes: exploring sparse, unstructured video collections". In: *ACM Transactions on Graphics (TOG)* 31.4 (2012), p. 68 (Cited on pages 80, 82, 113).

[125] Andreas Turina, Tinne Tuytelaars, and Luc Van Gool. "Efficient grouping under perspective skew". In: *CVPR*. Vol. 1. IEEE. 2001, pp. I–247 (Cited on page 53).

[126] Benjamin Ummenhofer and Thomas Brox. "Point-based 3d reconstruction of thin objects". In: *ICCV*. 2013, pp. 969–976 (Cited on page 55).

[127] Daniel Weinland, Edmond Boyer, and Remi Ronfard. "Action recognition from arbitrary views using 3d exemplars". In: *ICCV 2007-11th IEEE International Conference on Computer Vision*. IEEE. 2007, pp. 1–7 (Cited on pages 80, 81).

[128] Yonatan Wexler, Andrew Fitzgibbon, and Andrew Zisserman. "Bayesian estimation of layers from multiple images". In: *European Conference on Computer Vision*. Springer. 2002, pp. 487–501 (Cited on pages 54, 57, 61).

[129] Yonatan Wexler, Eli Shechtman, and Michal Irani. "Space-time completion of video". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 3 (2007), pp. 463–476 (Cited on pages 28, 29, 32, 33).

[130] Oliver Whyte, Josef Sivic, and Andrew Zisserman. "Get Out of my Picture! Internet-based Inpainting." In: *BMVC*. Vol. 2. 4. 2009, p. 5 (Cited on pages 29, 35, 36, 44, 46).

[131] Changchang Wu. "VisualSFM: A visual structure from motion system". In: (2011) (Cited on page 16).

[132] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M Seitz. "Multicore bundle adjustment". In: *CVPR 2011*. IEEE. 2011, pp. 3057–3064 (Cited on page 32).

[133] Changchang Wu, Jan-Michael Frahm, and Marc Pollefeys. "Detecting large repetitive structures with salient boundaries". In: *ECCV*. Springer. 2010, pp. 142–155 (Cited on pages 54, 75).

[134] Tianfan Xue, Michael Rubinstein, Ce Liu, and William T Freeman. "A computational approach for obstruction-free photography". In: *ACM Trans. on Graphics (TOG)* 34.4 (2015), p. 79 (Cited on pages 52, 54, 74, 75).

[135] Atsushi Yamashita, Akiyoshi Matsui, and Toru Kaneko. "Fence removal from multi-focus images". In: *ICPR*. IEEE. 2010, pp. 4532–4535 (Cited on page 54).

[136] Renjiao Yi, Jue Wang, and Ping Tan. "Automatic Fence Segmentation in Videos of Dynamic Scenes". In: *CVPR*. 2016, pp. 705–713 (Cited on pages 54, 74).

[137] Guangming Zang, Ramzi Idouchi, Ran Tao, Gilles Lubineau, Peter Wonka, and Wolfgang Heidrich. "Space-time tomography for continuously deforming objects". In: *ACM Transactions on Graphics (TOG)* 37.4 (2018), p. 100 (Cited on pages 80, 83).

[138] Cha Zhang and Tsuhan Chen. "A survey on image-based rendering—representation, sampling and compression". In: *Signal Processing: Image Communication* 19.1 (2004), pp. 1–28 (Cited on page 10).

[139] C Lawrence Zitnick and Sing Bing Kang. "Stereo for image-based rendering using image over-segmentation". In: *International Journal of Computer Vision* 75.1 (2007), pp. 49–65 (Cited on pages 28, 34).

[140] C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. "High-quality video view interpolation using a layered representation". In: *ACM transactions on graphics (TOG)*. Vol. 23. 3. ACM. 2004, pp. 600–608 (Cited on page 19).